REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	5.Sep.02	THESIS

4. TITLE AND SUBTITLE
FORMATION FLYING SATELLITE CONTROL AROUND THE L2 SUN-EARTH LIBRATH POINT

5. FUNDING NUMBERS

6. AUTHOR(S)
2D LT HAMILTON NICHOLAS H

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
GEORGE WASHINGTON UNIVERSITY

8. PERFORMING ORGANIZATION REPORT NUMBER
CI02-515

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

20021015 111

14. SUBJECT TERMS

15. NUMBER OF PAGES
165

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Title: Formation Flying Satellite Control Around the L2 Sun-Earth Libration Point

Author: Nicholas H. Hamilton

Rank/Service: Second Lieutenant, United States Air Force

Year: 2001

Number of pages: 165

Degree: Master of Science

Institution: The School of Engineering and Applied Science of the George Washington University

Abstract:

A growing interest in formation flying satellites demands development and analysis of control and estimation algorithms for station-keeping and formation maneuvering. This thesis discusses the development of a discrete linear-quadratic-regulator control algorithm for formations in the vicinity of the L2 sun-earth libration point. The development of an appropriate Kalman filter is included as well. Simulations are created for the analysis of the station-keeping and various formation maneuvers of the Stellar Imager mission. The simulations provide tracking error, estimation error, and control effort results. From the control effort, useful design parameters such as ΔV and propellant mass are determined. For formation maneuvering, the drone spacecraft track to within 4 meters of their desired position and within 1.5 millimeters per second of their desired zero velocity. The filter, with few exceptions, keeps the estimation errors within their three-sigma values. Without noise, the controller performs extremely well, with the drones tracking to within several micrometers. Each drone uses around 1 to 2 grams of propellant per maneuver, depending on the circumstances.

Bibliography:

1. Szebehely, V., *Theory of Orbits: The Restricted Problem of Three Bodies.* Academic Press, Inc. 1967.
2. Barden, B.T., and Howell, K.C., "Fundamental Motions Near Collinear Libration Points and Their Transitions," *Journal of the Astronautical Sciences*, Vol. 46. 1998.
3. Howell, K.C., Barden, B.T., and Lo, M.W., "Application of Dynamical Systems Theory to Trajectory Design for a Libration Point Mission," *Journal of the Astronautical Sciences*, Vol. 45. 1997.
4. Farquhar, R., *The Control and Use of Libration-Point Satellites.* NASA Technical Report R-346. 1970.
5. Gomez, G., Masdemont, J., and Simo, C., "Quasihalo Orbits Associated with Libration Points," *Journal of the Astronautical Sciences*, Vol. 46. 1998.
6. Scheeres, D.J., and Vinh, N.X., "Dynamics and Control of Relative Motion in an Unstable Orbit," AIAA Paper 2000-4135. 2000.

7. Hoffman, D.A., *Station-keeping at the Collinear Equilibrium Points of the Earth-Moon System*, JSC-26189. NASA Johnson Space Center. 1993.
8. "Microwave Anisotropy Probe," http://map.gsfc.nasa.gov. October, 2001.
9. "The GSFC Stellar Imager Homepage," http://hires.gsfc.nasa.gov/~si/. October, 2001.
10. "Micro Arcsecond X-Ray Imaging Mission," http://maxim.gsfc.nasa.gov. October, 2001.
11. "MAXIM Pathfinder," http://maxim.gsfc.nasa.gov/pathfinder.html. October, 2001.
12. "Terrestrial Planet Finder," http://tpf.jpl.nasa.gov. October, 2001.
13. Speyer, J.L., "Computation and Transmission Requirements for a Decentralized Linear-Quadratic-Gaussian Control Problem." *IEEE Transactions on Automatic Control*; AC-24(2). 1979.
14. Carpenter, J.R., "A Preliminary Investigation of Decentralized Control for Satellite Formations." Proceedings of the 2000 IEEE Aerospace Conference, March 18-25, 2000.
15. Carpenter, J.R., "Decentralized Control of Satellite Formations." To appear in *International Journal of Robust and Nonlinear Control.*
16. Carpenter, J.R., Folta, D.C., and Quinn, D.A., "Integration of Decentralized Linear-Quadratic-Gaussian Control into GSFC's Universal 3-D Autonomous Formation Flying Algorithm." AIAA Paper 99-4269, AIAA Guidance Navigation &Control, Modeling & Simulation Technologies and Atmospheric Flight Mechanics Conference and Exhibit, August 9-11, 1999.
17. "Magnetospheric Constellation," http://stp.gsfc.nasa.gov/missions/mc/mc.htm. October 2001.
18. "Laser Interferometer Space Antenna," http://lisa.jpl.nasa.gov. October 2001.
19. Wie, B., *Space Vehicle Dynamics and Control.* AIAA Education Series. 1998.
20. Nise, N., *Control Systems Engineering.* 2nd Edition. Addison-Wesley Publishing Company. 1995.
21. Friedland, B., *Control System Design.* McGraw-Hill. 1986.
22. Stengel, R.F., *Optimal Control and Estimation.* Dover Publications, Inc. 1994.
23. Phillips, C.L., and Nagle, H.T., *Digital Control System Analysis and Design.* 3rd Edition. Prentice Hall, Inc. 1995.
24. Bryson, A.E, Jr., and Ho, Y.C., *Applied Optimal Control: Optimization, Estimation, and Control.* Hemisphere Publishing Corporation. 1975.
25. Wagner, C., "Formation Flying Around Sun-Earth Libration Point L1." International Space University. 2000.
26. Brown, R.C., and Hwang, P.Y.C., *Introduction to Random Signals and Applied Kalman Filtering.* 2nd Edition. John Wiley & Sons, Inc. 1992.
27. Maybeck, P.S., *Stochastic Models, Estimation, and Control.* Volume 1. Academic Press, Inc. 1979.
28. Carpenter, K.G., Neff, S.G., Schrijver, C.J., Allen, R.J., and Rajagopal, J., "The Stellar Imager (SI) Mission Concept." In proceedings of the 36th Liege Astrophysical Colloquium: From Optical to Millimetric Intefereometry: Scientific and Technical Challenges. 2001.
29. Carpenter, K.G., Lyon, R.G., Schrijver, C.J., Mundy, L.J., Allen, R.J., and Rajagopal, J., "Imaging the Surfaces and Interiors of Other Stars: The Stellar Imager (SI) Mission Concept." 2001.

30. Rarogiewicz, "Introduction to Interferometry." http://www.mtwilson.edu/Education/Presentations/Interferometry/. October 2001.
31. Golomb, S.W., and Taylor, H. IEEE Trans. Info. Theo., 28, #4. 1982.
32. Leitner, J., and Schnurr, R., "Stellar Imager (SI): Formation Flying." Presentation from the Integrated Mission Design Center, Goddard Space Flight Center, NASA. 2001.
33. US Government Printing Office, *The Astronomical Almanac for the Year 2000.* 1998.
34. Howell, K.C., and Anderson, J., *Generator User's Guide.* Version 3.0.2. 2001.
35. Anderson, J., "Earth-to-Lissajous Transfers and Recovery Trajectories Using an Ephemeris Model." Purdue University. 2001.
36. Humble, R.W., Henry, G.N., and Wiley, J.L., *Space Propulsion Analysis and Design.* The McGraw-Hill Companies, Inc. 1995.
37. Asato, D., "Stellar Imager: Propulsion." Presentation from the Integrated Mission Design Center, Goddard Space Flight Center, NASA. 2001.

Formation Flying Satellite Control Around the L2 Sun-Earth Libration Point

By

2nd Lieutenant Nicholas H. Hamilton

B.S. May, 2000, United States Air Force Academy

A Thesis submitted to

The Faculty of

The School of Engineering and Applied Science
of the George Washington University in partial satisfaction
of the requirements for the degree of Master of Science

19 December 2001

Thesis directed by

Catherine Mavriplis

Professor of Engineering and Applied Science

THE VIEWS EXPRESSED IN THIS ARTICLE ARE THOSE OF THE AUTHOR AND DO NOT REFLECT THE OFFICIAL POLICY OR POSITION OF THE UNITED STATES, DEPARTMENT OF DEFENSE, OR THE U.S. GOVERNMENT

ABSTRACT

A growing interest in formation flying satellites demands development and analysis of control and estimation algorithms for station-keeping and formation maneuvering. This thesis discusses the development of a discrete linear-quadratic-regulator control algorithm for formations in the vicinity of the L2 sun-earth libration point. The development of an appropriate Kalman filter is included as well. Simulations are created for the analysis of the station-keeping and various formation maneuvers of the Stellar Imager mission. The simulations provide tracking error, estimation error, and control effort results. From the control effort, useful design parameters such as ΔV and propellant mass are determined. For formation maneuvering, the drone spacecraft track to within 4 meters of their desired position and within 1.5 millimeters per second of their desired zero velocity. The filter, with few exceptions, keeps the estimation errors within their three-sigma values. Without noise, the controller performs extremely well, with the drones tracking to within several micrometers. Each drone uses around 1 to 2 grams of propellant per maneuver, depending on the circumstances.

CONTENTS

LIST OF PROMINENT EQUATIONS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	Dynamics matrix
A_d	Discrete dynamics matrix
a_i	Rotating coordinate system basis vectors
az	Azimuth
B	Control-mapping matrix
B_d	Discrete control-mapping matrix
$\mathbf{b}_i$	Drone-centered coordinate system basis vectors
C	Measurement-mapping matrix
D	Distance between two massive bodies
D_i	Distance from system barycenter to respective bodies
el	Elevation
F_g	Gravitational force
G	Universal gravitation constant
H	Hamiltonian function, measurement-mapping matrix
I	Identity matrix
J	Cost function
K	Control gain
K_d	Discrete control gain
L	Kalman gain
M_d	State-control coupling weighting matrix
m_0	Initial mass of spacecraft
m_i	Masses of respective bodies
m_{prop}	Propellant mass
$\hat{n}_i$	Inertial coordinate system basis vectors
$P(-)$	Estimation error covariance matrix before measurements
$P(+)$	Estimation error covariance matrix after measurements
p	Position vector of spacecraft with respect to the libration point
Q	Process noise covariance
Q_c	Continuous process noise strength
R	Measurement noise covariance
r	Position vector of spacecraft in rotating frame, radial spherical coordinate
$\mathbf{r}$	Range
r_0	Position vector of libration point
r_i	Position vector of spacecraft from respective body
S	Co-state to state transition matrix
T	Sampling interval, maneuver interval
U	Potential function
$\mathbf{u}$	Control vector
V	Control weighting matrix

V_d	Discrete control weighting matrix
v	Measurement noise vector
W	State tracking error weighting matrix
W_d	Discrete state tracking error weighting matrix
w	Process noise vector
w_c	Continuous system process noise vector
X	Radial coordinate in rotating frame
x	Radial coordinate in libration point-centered, Earth-centered, or hub-centered frames
$\mathbf{x}$	State vector
$\tilde{\mathbf{x}}$	State tracking error vector
$\hat{\mathbf{x}}^-$	State tracking error *a priori* estimate vector
$\hat{\mathbf{x}}$	State tracking error *a posteriori* estimate vector
$\bar{\mathbf{x}}^-$	State *a priori* estimate vector
$\mathbf{x}^{ref}$	State reference vector
Y	Along-track coordinate in rotating frame
y	Along-track coordinate in libration point-centered, Earth-centered, or hub-centered frames
$\mathbf{y}$	Measurement vector
$\hat{\mathbf{y}}$	Measurement estimate vector
Z	Cross-track coordinate in rotating frame
z	Cross-track coordinate in libration point-centered, Earth-centered, or hub-centered frames
ΔV	Velocity change
Φ	State-transition matrix
Φ_{cl}	Closed-loop state-transition matrix
ϕ	Second angular spherical coordinate
λ_i	Eigenvalue, co-state vector
μ_i	Gravitation constant of respective bodies
θ	First angular spherical coordinate
σ	Estimation error standard deviation
ω	Angular velocity of 2-body system
ω_{xy}	In-plane frequency
ω_z	Out-of-plane frequency
$\Im$	Lyapunov function

1. INTRODUCTION

1.1 Libration Point Introduction

Discovered by Euler and Lagrange in 1772, while studying the motion of the moon, libration points are a significant niche in the field of orbital dynamics. In the simplest sense, libration points are points in space where the gravitational and centrifugal forces cancel. Szebehely[1] studied the mathematics of the generic restricted three-body problem, and formulated the positions and behaviors of the libration points and the dynamics of objects in their vicinity. Barden[2], and later Howell[3], expanded the study of libration point dynamics to their transitions and trajectory design.

Farquhar[4] initially examined control techniques for the station-keeping of satellites orbiting a libration point. Gomez, Masdemont, and Simo[5] have extensively studied dynamics and control problems involving libration points. Scheeres and Vinh[6] have researched the dynamics and control of relative motion of two spacecraft in unstable orbits. Hoffman[7] used modern control techniques to perform station keeping around the earth-moon collinear libration points.

Many future space missions are planning to use libration points. In fact, the Microwave Anisotropy Probe[8] (MAP) began orbiting the sun-earth L2 point on 1 October 2001. Other future libration point missions include Stellar Imager[9], Micro Arcsecond X-Ray Interferometry Mission[10] (MAXIM), MAXIM Pathfinder[11], and possibly Terrestrial Planet Finder[12]. Further understanding of the dynamics and control of satellites in the vicinity of libration points is essential for the success of these and many other future space missions.

1.2 Formation Flying Introduction

Currently, formation flying spacecraft control is being extensively researched. Flying satellites relative to one another offers mission possibilities otherwise infeasible. Multiple satellites working together allows for more powerful telescopes, interferometers, and other scientific data collecting instruments. Communications technology may improve as well. With multiple satellites, risk is reduced through redundancy, so a failure of one element does not compromise the entire mission. Also, if the spacecraft are similar or identical, assembly line production can reduce costs.

Formation flying control can be performed in two ways--centralized or decentralized. With centralized control, one spacecraft or processor calculates and commands the motion of the entire formation. With decentralized control, each spacecraft, with input from the rest of the formation, processes its own control requirements. Although this thesis presents a centralized control approach, the decentralized method is the basis for many references. Speyer[13] first introduced a decentralized linear-quadratic-Gaussian control method. Carpenter[14, 15] applied this work to formation flying satellites, and further expanded it to deal with both time-invariant and time-varying systems. Speyer's[13] method produces identical results to the centralized linear-quadratic-Gaussian control method, and it also minimizes data transmission.

Formation flying satellites in Earth orbit have been demonstrated. Carpenter, Folta, and Quinn[16] derived a decentralized framework for the applicability of autonomous formation flying control for the EO-1 mission to follow Landsat-7 in low-Earth orbit.

The next step is extending formation flying missions to other points in space, such as the sun-earth libration points.

NASA has several distributed spacecraft missions planned for the next decade and beyond. The Magnetospheric Constellation[17] will study the magnetotail of the earth. Stellar Imager[9] and MAXIM[10] will image stars and black holes respectively. Also, the Laser Interferometer Space Antenna[18] will detect gravitational waves in an attempt to prove elements of general relativity. All of these missions, and many more, rely on formation flying.

1.3 Overview

In Section 2, the equations of motion for the circular restricted-three body problem are developed and explained. From these equations of motion, the libration points are discovered and addressed. The problem is simplified by linearizing the equations of motion about the collinear libration points. Then, the orbits unique to libration points are examined.

In Section 3, the open-loop dynamics of a system are briefly explained. State feedback control is applied to "close the loop." The optimal control is found using the linear-quadratic-regulator method. Then, the problem is simplified with an infinite horizon assumption. The stability of the linear-quadratic-regulator is proven. In Section 4, the continuous system is sampled, and the discrete optimal control law is found. These concepts are then applied to satellites flying in formation in Section 5.

In Section 6, an observer is introduced to estimate the states of a system by using incoming measurements. Augmenting the system state-space for multiple satellites is

shown. The optimal method of updating the estimates is given, and the Kalman filter is derived. A common Kalman filter algorithm is shown. The Kalman filter is adapted to accommodate nonlinear measurements, with range, azimuth, and elevation given as an example.

The background to the Stellar Imager mission is explored in Section 7, yielding performance and design requirements. Special emphasis is given to the preliminary formation, orbit, and control designs. In Section 8, the reference orbit and dynamics specific to the sun-earth L2 point are determined from the circular restricted three-body; however, a more realistic orbit and associated dynamics are simulated instead. Simulations are developed to maintain a satellite on this desired reference orbit, slew the formation to aim at another star, and reconfigure the formation to achieve better imaging results. The simulations produce useful tracking error, estimation error, and control effort results.

2. CIRCULAR RESTRICTED THREE-BODY PROBLEM

2.1 Equations of Motion

The classic restricted three-body problem refers to the motion of an infinitesimally small body in the presence of two larger bodies orbiting their barycenter, or common center of mass. Two examples of the restricted three-body problem are the earth-moon and sun-earth systems. For the earth-moon system, the earth is the first body, the moon, orbiting the earth, is the second body, and a satellite would be the third. For the sun-earth system, the sun is the first body, the earth, orbiting the sun, is the second body, and a satellite would be third. For preliminary analysis, the orbits of the moon around the earth and the earth around the sun are assumed to be circular, have a constant angular velocity, and have no variation in inclination from the orbital plane. When these assumptions are made, the restricted three-body problem becomes the more specific circular restricted three-body problem. This thesis will focus on the sun-earth circular restricted three-body problem, although the development of equations will hold for any three bodies.

For the development of the equations of motion, the rotating coordinate system shown in Figure 1 will be used. The origin is at the barycenter of the two large bodies. The first basis vector, a_1, runs along the line between the two large bodies in the direction of the smaller body. The third basis vector, a_3, is in the direction of the orbit normal. The cross product of a_3 and a_1 yields the second basis vector, a_2. These vectors form an orthogonal coordinate system.

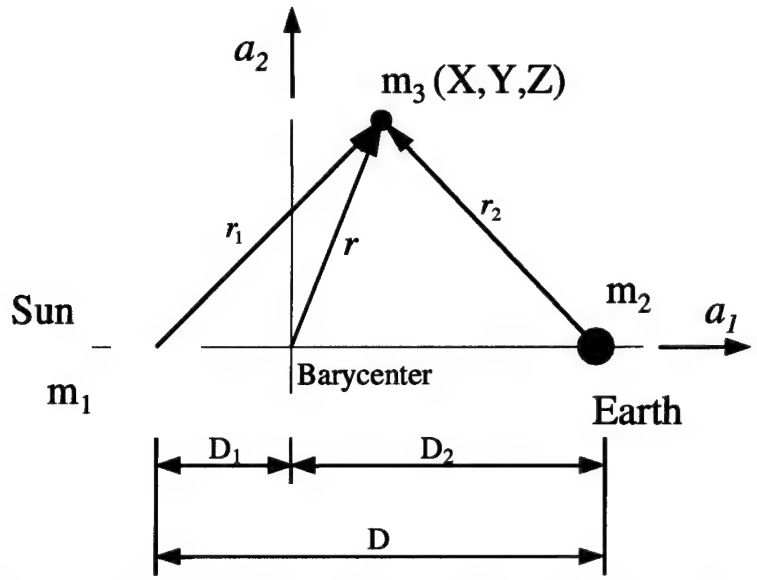

Figure 1. The three-body problem rotating coordinate system

The position vector of the spacecraft in the rotating frame is

$$r = Xa_1 + Ya_2 + Za_3 . \tag{2.1}$$

Writing the vectors from the large bodies to the spacecraft in the rotating frame gives

$$r_1 = (X + D_1)a_1 + Ya_2 + Za_3 \tag{2.2}$$

and

$$r_2 = (X - D_2)a_1 + Ya_2 + Za_3 , \tag{2.3}$$

where D_1 is the distance from the barycenter to the largest body and D_2 is the distance from the barycenter to the smaller body. Note that

$$D = D_1 + D_2 . \tag{2.4}$$

An inertial coordinate system is defined such that the origin is the barycenter, but the basis vectors do not rotate with the system. The inertial basis vectors $\hat{n}_1$ and $\hat{n}_2$ lie in the same plane as the rotating basis vectors a_1 and a_2; thus, $\hat{n}_3$ is identical to a_3. Once every revolution $\hat{n}_1$ is aligned with a_1, and $\hat{n}_2$ is aligned with a_2. With respect to the

inertial coordinate frame, the two large bodies are rotating about their barycenter with a constant angular velocity

$$\bar{\omega} = [0 \quad 0 \quad \omega]^T, \tag{2.5}$$

where

$$\omega = \sqrt{\frac{G(m_1 + m_2)}{D^3}}. \tag{2.6}$$

Therefore, deriving with respect to the inertial frame

$$\dot{a}_1 = \omega a_2 \tag{2.7}$$

$$\dot{a}_2 = -\omega a_1 \tag{2.8}$$

$$\dot{a}_3 = 0. \tag{2.9}$$

Taking the first and second time derivatives of the position vector (Equation 2.1) with respect to the inertial frame,

$$\dot{r} = (\dot{X} - \omega Y)a_1 + (\omega X + \dot{Y})a_2 + \dot{Z}a_3 \tag{2.10}$$

and

$$\ddot{r} = (\ddot{X} - 2\omega\dot{Y} - \omega^2 X)a_1 + (\ddot{Y} + 2\omega\dot{X} - \omega^2 Y)a_2 + \ddot{Z}a_3. \tag{2.11}$$

Newton's Law of Universal Gravitation for n bodies states

$$F_g = -Gm_i \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{m_j}{|r_{ji}|^3} r_{ji}, \tag{2.12}$$

where F_g is the total gravitational force acting on the i^{th} body, G is the universal gravitation constant, m represents the masses of the respective bodies, and r_{ji} is the position vector from the j^{th} body to the i^{th} body. Newton's Second Law defines force as the time derivative of momentum

$$F = m_i \frac{dv_i}{dt} + v_i \frac{dm_i}{dt}, \tag{2.13}$$

where

$$\dot{r}_i = v_i = \frac{dr_i}{dt}, \tag{2.14}$$

and

$$\ddot{r}_i = \frac{dv_i}{dt} = \frac{d^2 r_i}{dt^2}. \tag{2.15}$$

The derivatives in Equations 2.14 and 2.15 are with respect to an inertial frame.

Assuming constant mass, then

$$\frac{dm_i}{dt} = 0. \tag{2.16}$$

With gravity as the only force (the restricted part of the restricted three-body problem), substituting Equations 2.13 and 2.16 into Equation 2.12, and dividing by m_i, yields

$$\ddot{r}_i = -G \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{m_j}{\left| r_{ji} \right|^3} r_{ji}. \tag{2.17}$$

Expanding for the three-body problem and dropping the subscript i gives the equation of motion for a satellite in the restricted three-body problem

$$\ddot{r} = -\frac{Gm_1}{\left| r_1 \right|^3} r_1 - \frac{Gm_2}{\left| r_2 \right|^3} r_2. \tag{2.18}$$

Substituting Equations 2.11, 2.2, and 2.3 into Equation 2.18, the equations of motion can now be written as

$$\ddot{X} - 2\omega\dot{Y} - \omega^2 X = -\frac{\mu_1 (X + D_1)}{\left| r_1 \right|^3} - \frac{\mu_2 (X - D_2)}{\left| r_2 \right|^3} \tag{2.19}$$

$$\ddot{Y} + 2\omega\dot{X} - \omega^2 Y = -\frac{\mu_1 Y}{|r_1|^3} - \frac{\mu_2 Y}{|r_2|^3} \tag{2.20}$$

$$\ddot{Z} = -\frac{\mu_1 Z}{|r_1|^3} - \frac{\mu_2 Z}{|r_2|^3}, \tag{2.21}$$

where

$$\mu_j = Gm_j. \tag{2.22}$$

Szebehely[1] shows that the centrifugal plus gravitational force potential, U, exists such that

$$U = \frac{1}{2}\omega^2(X^2 + Y^2) + \frac{\mu_1}{|r_1|} + \frac{\mu_2}{|r_2|}, \tag{2.23}$$

and

$$\ddot{X} - 2\omega\dot{Y} = \frac{\partial U}{\partial X} \tag{2.24}$$

$$\ddot{Y} + 2\omega\dot{X} = \frac{\partial U}{\partial Y} \tag{2.25}$$

$$\ddot{Z} = \frac{\partial U}{\partial Z}. \tag{2.26}$$

2.2 Libration Points

By setting all time derivatives in the equations of motion to zero, five libration, or Lagrange, points can be calculated. The location of these points depend on the masses and distances of the bodies; however, three points are always collinear with the two large bodies (L1, L2, and L3), and the other two points form equilateral triangles with the two large bodies (L4, and L5). Szebehely[1] calls the point between the two masses L2 and the point opposite the smaller body L1. However, most of today's literature, including Wie[19]

and Farquhar[4], have just the opposite, with L1 being between the two masses and L2 opposite the smaller body, as shown in Figure 2.

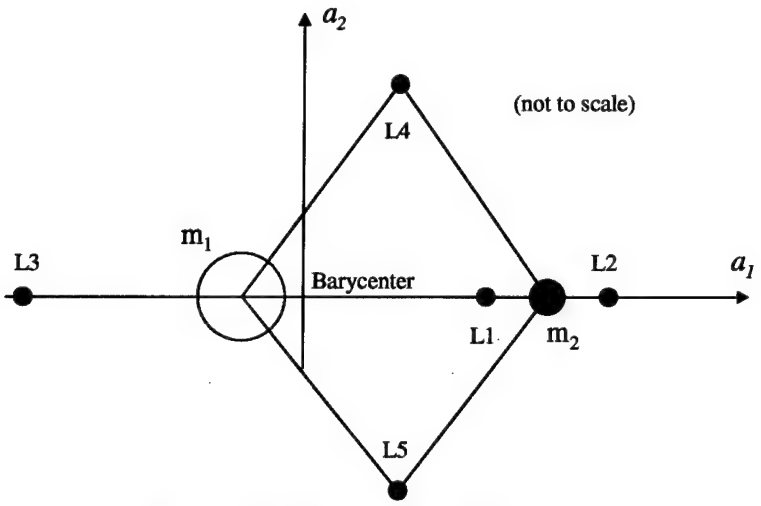

Figure 2. Libration point locations

For the sun-earth system, it is common to use the earth-moon barycenter as the second body rather than the earth itself, and is done so by Szebehely[1]. Hoffman[7] shows that the motion near the equilateral libration points is stable when the ratio of the smaller body mass to the total system mass is less than 0.0385, and that the motion near the collinear libration points is always unstable. Barring any other forces, an object's orbit around a stable equilateral libration point will eventually decay until it rests at the libration point itself. Szebehely[1] mentions the Trojan asteroids sit at a sun-Jupiter equilateral libration point. An object in an orbit around a collinear libration point, on the other hand, will eventually fly away from the vicinity of the libration point.

2.3 Linearized Equations of Motion Near Collinear Libration Points

Motion around a collinear libration point is more easily expressed in a local coordinate frame, shown in Figure 3, with the origin located at the libration point. The

radial direction, x, is collinear with a_1, going from the sun (or largest body) through the libration point. The cross-track direction, z, is parallel and in the same direction as a_3, the orbit normal. Crossing z with x gives the in-track (or along-track) direction, y. When circular motion is assumed, the y direction is the tangential velocity direction of the libration point around the system barycenter. These axes form an orthogonal coordinate system. Motion in the x-y plane will be referred to as in-plane, and any motion in the z direction will be referred to as out-of-plane.

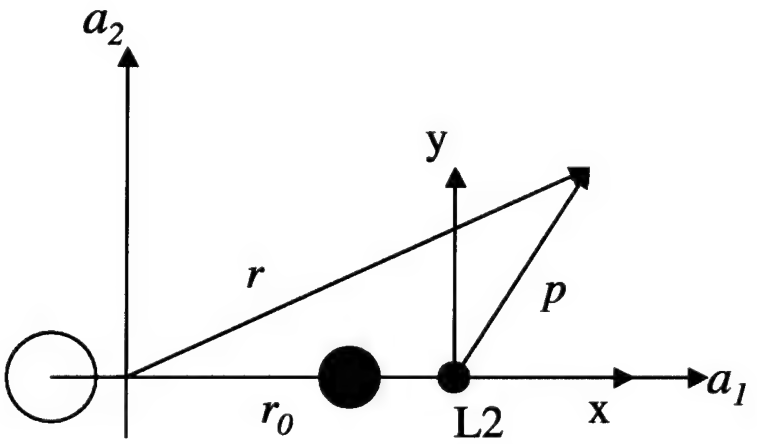

Figure 3. Relative position with respect to L2

Next, break the position vector, r, into the libration point position vector, r_0, and the vector from the libration point to the satellite, p, where

$$r = r_0 + p, \tag{2.27}$$

and p is expressed in the local coordinate frame. The individual coordinates can be written as

$$X = X_0 + x \tag{2.28}$$

$$Y = Y_0 + y \tag{2.29}$$

$$Z = Z_0 + z, \tag{2.30}$$

where X_0, Y_0, and Z_0 are the coordinates of the libration point.

So far, the equations of motion contain nonlinear terms, all stemming from the potential, U. To simplify for continued analysis, these equations must be linearized. To do so, substitute Equations 2.28-2.30 into Equations 2.24-2.26 and perform a Taylor series expansion on the partials of U, about the libration point. The series is truncated after the quadratic terms, and the derivatives are evaluated at the libration point. This gives equations of motion with respect to the libration point in the local coordinate frame:

$$\ddot{x} - 2\omega\dot{y} - U_{XX}x - U_{XY}y = 0 \tag{2.31}$$

$$\ddot{y} + 2\omega\dot{x} - U_{YY}y - U_{XY}x = 0 \tag{2.32}$$

$$\ddot{z} - U_{ZZ}z = 0, \tag{2.33}$$

where

$$U_{XX} = \left.\frac{\partial^2 U}{\partial X^2}\right|_{r_0}, \tag{2.34}$$

$$U_{YY} = \left.\frac{\partial^2 U}{\partial Y^2}\right|_{r_0}, \tag{2.35}$$

$$U_{ZZ} = \left.\frac{\partial^2 U}{\partial Z^2}\right|_{r_0}, \tag{2.36}$$

and

$$U_{XY} = \left.\frac{\partial^2 U}{\partial X \partial Y}\right|_{r_0}. \tag{2.37}$$

The x and y motions are coupled, and the z motion is independent. Therefore, all cross-term derivatives involving z disappear and are not shown above. Also, for collinear libration points,

$$U_{XY} = 0. \tag{2.38}$$

The linearized equations of motion about a collinear libration point become

$$\ddot{x} - 2\omega\dot{y} - U_{XX}x = 0 \tag{2.39}$$

$$\ddot{y} + 2\omega\dot{x} - U_{YY}y = 0 \tag{2.40}$$

$$\ddot{z} - U_{ZZ}z = 0. \tag{2.33}$$

2.4 Lissajous Orbits and Quasi-Periodic Trajectories

By building a state vector, **x**, consisting of the positions and velocities, the equations of motion can be expressed in state-space notation as

$$\dot{\mathbf{x}} = A\mathbf{x}, \tag{2.41}$$

where

$$\mathbf{x} = [x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z}]^T, \tag{2.42}$$

$$\dot{\mathbf{x}} = [\dot{x} \quad \dot{y} \quad \dot{z} \quad \ddot{x} \quad \ddot{y} \quad \ddot{z}]^T, \tag{2.43}$$

and

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ U_{XX} & 0 & 0 & 0 & 2\omega & 0 \\ 0 & U_{YY} & 0 & -2\omega & 0 & 0 \\ 0 & 0 & U_{ZZ} & 0 & 0 & 0 \end{bmatrix}. \tag{2.44}$$

There exists a quasi-periodic solution to the linearized equations of motion. The development here is very similar to that in Wie[19]. Because the z motion is uncoupled, two characteristic equations exist, the in-plane equation,

$$\lambda^4 + (4\omega - U_{XX} - U_{YY})\lambda^2 + U_{XX}U_{YY} = 0 \tag{2.45}$$

and the out-of-plane equation,

$$\lambda^2 - U_{zz} = 0. \tag{2.46}$$

The out-of-plane equation is a simple harmonic oscillator. The eigenvalues are

$$\lambda_{5,6} = \pm j\sqrt{|U_{zz}|} = \pm j\omega_z, \tag{2.47}$$

where ω_z is the out-of-plane frequency. Out-of-plane motion is always periodic. The in-plane eigenvalues are

$$\lambda_{1,2} = \pm\sqrt{-\beta_1 + \sqrt{\beta_1^2 + \beta_2^2}}, \tag{2.48}$$

and

$$\lambda_{3,4} = \pm j\sqrt{\beta_1 + \sqrt{\beta_1^2 + \beta_2^2}} = \pm j\omega_{xy}, \tag{2.49}$$

where ω_{xy} is the in-plane frequency,

$$\beta_1 = 2\omega - \frac{(U_{XX} + U_{YY})}{2}, \tag{2.50}$$

and

$$\beta_2^2 = -U_{XX}U_{YY}. \tag{2.51}$$

Szebehely[1] proves that U_{XX} is always positive, whereas U_{YY} and U_{ZZ} are always negative. The in-plane motion has two oscillatory poles, one stable pole, and one unstable pole. By judiciously choosing initial conditions, the stable and unstable poles can be made to vanish. Let

$$Q = [q_1 \quad \cdots \quad q_6], \tag{2.52}$$

and

$$P = [p_1 \quad \cdots \quad p_6], \tag{2.53}$$

where q_i is the normalized right-side eigenvector corresponding to λ_i and p_i is the normalized left-side eigenvector corresponding to λ_i, then through modal decomposition

$$A = Q\Lambda P^T \tag{2.54}$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_6 \end{bmatrix} \tag{2.55}$$

and

$$P^T Q = I\,. \tag{2.56}$$

The solution to the differential equation in Equation 2.40 then becomes

$$\mathbf{x}(t) = e^{At}\mathbf{x}(0) = \sum_{i=1}^{6} e^{\lambda_i t} q_i p_i^T \mathbf{x}(0)\,. \tag{2.57}$$

Note that $p_i^T \mathbf{x}(0)$ is scalar. Setting

$$p_1^T \mathbf{x}(0) = p_2^T \mathbf{x}(0) = 0 \tag{2.58}$$

leaves only oscillatory poles remaining. These conditions can be met if

$$\dot{x}(0) = \frac{\omega_{xy}}{\kappa} y(0), \tag{2.59}$$

and

$$\dot{y}(0) = -\kappa\omega_{xy} x(0), \tag{2.60}$$

where

$$\kappa = \frac{\omega_{xy}^2 + U_{XX}}{2\omega\omega_{xy}}. \tag{2.61}$$

Finally, the quasi-periodic solution to the linearized equations of motion for a collinear libration point can be expressed simply as

$$x(t) = x(0)\cos(\omega_{xy}t) + \frac{y(0)}{\kappa}\sin(\omega_{xy}t) \tag{2.62}$$

$$y(t) = y(0)\cos(\omega_{xy}t) - \kappa x(0)\sin(\omega_{xy}t) \tag{2.63}$$

$$z(t) = z(0)\cos(\omega_z t) + \frac{\dot{z}(0)}{\omega_z}\sin(\omega_z t). \tag{2.64}$$

Orbits of this type are called Lissajous trajectories. A Lissajous orbit is quasi-periodic because the ratio of in-plane frequency to out-of-plane frequency is not an integer. Barring any forces other than the gravity of the two large bodies, a satellite with the proper initial conditions in the circular restricted three-body setting will follow a Lissajous orbit forever.

3. LINEAR QUADRATIC REGULATOR THEORY

3.1 Open-Loop Dynamics

With no control, disturbances, or perturbations, and proper initial conditions, a satellite will follow the Lissajous orbit described above. This open-loop trajectory is useful as a preliminary reference trajectory for a spacecraft. Generally, the open-loop dynamic equation

$$\dot{\mathbf{x}}_{ref}(t) = A\mathbf{x}_{ref}(t) \tag{3.1}$$

has the solution

$$\mathbf{x}_{ref}(t) = \Phi(t-t_0)\mathbf{x}_{ref}(t_0) = e^{A(t-t_0)}\mathbf{x}_{ref}(t_0), \tag{3.2}$$

if A is a constant matrix. The vector $\mathbf{x}_{ref}(t)$ is the desired reference state at time t, and $\Phi(t-t_0)$ is the open-loop state transition matrix. It is shown by Nise[20] that the state transition matrix is easily approximated by an infinite series if A is constant,

$$\Phi(t-t_0) = e^{A(t-t_0)} = \sum_{k=0}^{\infty} \frac{A^k (t-t_0)^k}{k!}. \tag{3.3}$$

3.2 State Feedback Control

Perturbations, natural, man-made, or deliberate, will require the satellite be controlled. With the added control, the linear differential equation becomes

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \tag{3.4}$$

where $\mathbf{u}$ is the control vector and B is the matrix that maps the control effort to the state-space. The initial condition $\mathbf{x}(t_0)$ is also given. For this thesis, the system will remain

time-invariant (A and B are constant matrices). The control is modeled as ideally applied acceleration in the x, y, and z directions. Therefore,

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

and

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \tag{3.6}$$

where u_1 is the control in the x direction, u_2 is the control in the y direction, and u_3 is the control in the z direction. The control vector, $\mathbf{u}$, has the linear state feedback form

$$\mathbf{u} = -K(t)\mathbf{x}, \tag{3.7}$$

where K is a matrix of gains that relate the states to the control. This is also known as the control law. Closing the loop and substituting back into the differential equation yields

$$\dot{\mathbf{x}} = (A - BK(t))\mathbf{x}, \tag{3.8}$$

which has the solution

$$\mathbf{x}(t) = \Phi_{cl}(t, t_0)\mathbf{x}(t_0), \tag{3.9}$$

where $\Phi_{cl}(t, t_0)$ is the time-varying, closed-loop state-transition matrix. The state transition matrix propagates the state from some time (in this case t_0) to another time (in this case t). The state error is

$$\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_{ref}(t). \tag{3.10}$$

By differentiating Equation 3.10, it can be shown that $\tilde{\mathbf{x}}(t)$ follows the same dynamic constraint as the state. By superposition,

$$\dot{\tilde{\mathbf{x}}} = (A - BK(t))\tilde{\mathbf{x}}, \tag{3.11}$$

and

$$\tilde{\mathbf{x}}(t) = \Phi_{cl}(t, t_0)\tilde{\mathbf{x}}(t_0). \tag{3.12}$$

Also, the control law becomes

$$\mathbf{u} = -K(t)\tilde{\mathbf{x}}. \tag{3.13}$$

3.3 Continuous Optimal Control

The cost function

$$J = \int_{t_0}^{t_f} \left\{ \tilde{\mathbf{x}}^T(\tau) W \tilde{\mathbf{x}}(\tau) + \mathbf{u}^T(\tau) V \mathbf{u}(\tau) \right\} d\tau \tag{3.14}$$

is a scalar number that represents the desired performance of the system. The first part of the cost function deals with the state error. The matrix, W, weights the state error and must be positive semi-definite. The second part of the cost function deals with the control. The matrix, V, weights the control and must be positive definite and thus invertible. Altering the matrices W and V changes the interaction between the control and state error. For this thesis, no weight will be given to the coupling between states, the coupling between controls, or state-control coupling. Hence, W and V will remain diagonal.

The following development comes from Friedland[21]. The cost function can also be written in the form

$$J = \int_{t_0}^{t_f} \begin{bmatrix} \tilde{\mathbf{x}}^T(\tau) & \mathbf{u}^T(\tau) \end{bmatrix} \begin{bmatrix} W & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}(\tau) \\ \mathbf{u}(\tau) \end{bmatrix} d\tau . \tag{3.15}$$

The quadratic nature of the cost function ensures that it is always non-negative. Optimally, both the state error and control would be zero, as would J. Therefore, the purpose of control is to minimize the cost function. Substituting the control law, Equation 3.13, and the state-transition matrix, Equation 3.14, into the cost function gives

$$\begin{aligned} J = \int_{t_0}^{t_f} \{ & \tilde{\mathbf{x}}^T(t_0) \Phi_{cl}^T(\tau, t_0) W \Phi_{cl}(\tau, t_0) \tilde{\mathbf{x}}(t_0) \\ & + \tilde{\mathbf{x}}^T(t_0) \Phi_{cl}^T(\tau, t_0) K^T(\tau) V K(\tau) \Phi_{cl}(\tau, t_0) \tilde{\mathbf{x}}(t_0) \} d\tau \end{aligned} \tag{3.16}$$

Extracting the initial conditions from both sides of the integral and compressing,

$$J = \tilde{\mathbf{x}}^T(t_0) \left[\int_{t_0}^{t_f} \Phi_{cl}^T(\tau, t_0) \left[W + K^T(\tau) V K(\tau) \right] \Phi_{cl}(\tau, t_0) d\tau \right] \tilde{\mathbf{x}}(t_0) . \tag{3.17}$$

Next, define

$$S(t_0, t_f) = \int_{t_0}^{t_f} \Phi_{cl}^T(\tau, t_0) \left[W + K^T(\tau) V K(\tau) \right] \Phi_{cl}(\tau, t_0) d\tau , \tag{3.18}$$

so

$$J = \tilde{\mathbf{x}}^T(t_0) S(t_0, t_f) \tilde{\mathbf{x}}(t_0) . \tag{3.19}$$

Since the initial time, t_0, is arbitrary, then it holds for any time t that

$$J = \tilde{\mathbf{x}}^T(t) S(t, t_f) \tilde{\mathbf{x}}(t) . \tag{3.20}$$

It also holds that

$$J = \int_{t}^{t_f} \tilde{\mathbf{x}}^T(\tau) \left[W + K^T(\tau) V K(\tau) \right] \tilde{\mathbf{x}}(\tau) d\tau . \tag{3.21}$$

Using the integral-derivative theorem

$$\frac{d}{dx}\left\{\int_{A(x)}^{B(x)} F(x,\xi)d\xi\right\} = \int_{A(x)}^{B(x)} \frac{\partial F(x,\xi)}{\partial x}d\xi - F(x,A(x))\frac{dA(x)}{dx} + F(x,B(x))\frac{dB(x)}{dx}, \quad (3.22)$$

then

$$\frac{dJ}{dt} = -\tilde{\mathbf{x}}^T(t)\left[W + K^T(t)VK(t)\right]\tilde{\mathbf{x}}(t). \quad (3.23)$$

Also, by differentiating Equation 3.20,

$$\frac{dJ}{dt} = \dot{\tilde{\mathbf{x}}}^T(t)S(t,t_f)\tilde{\mathbf{x}}(t) + \tilde{\mathbf{x}}^T(t)\dot{S}(t,t_f)\tilde{\mathbf{x}}(t) + \tilde{\mathbf{x}}^T(t)S(t,t_f)\dot{\tilde{\mathbf{x}}}(t), \quad (3.24)$$

which reduces to

$$\frac{dJ}{dt} = \tilde{\mathbf{x}}^T(t)\left[(A-BK(t))^T S(t,t_f) + \dot{S}(t,t_f) + S(t,t_f)(A-BK(t))\right]\tilde{\mathbf{x}}(t). \quad (3.25)$$

Setting Equation 3.23 equal to Equation 3.25,

$$-\left(W + K^T(t)VK(t)\right) = (A-BK(t))^T S(t,t_f) + \dot{S}(t,t_f) + S(t,t_f)(A-BK(t)), \quad (3.26)$$

or

$$-\dot{S}(t,t_f) = (A-BK(t))^T S(t,t_f) + S(t,t_f)(A-BK(t)) + W + K^T(t)VK(t). \quad (3.27)$$

The end condition

$$S(t_f,t_f) = 0 \quad (3.28)$$

may be used.

Next, we wish to find the gain matrix, $K(t)$, which minimizes the cost function, J. Assume that some optimal gain, $K^*(t)$, corresponds to the minimum cost function, J^*. Assume some other gain, $K^* + \Delta K$, corresponds to $S^* + \Delta S$, where ΔS is necessarily positive semi-definite. For clarity, all time indices have been removed for the time being. By proving that

$$J^* = \tilde{\mathbf{x}}^T S^* \tilde{\mathbf{x}} < \tilde{\mathbf{x}}^T (S^* + \Delta S)\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T S^* \tilde{\mathbf{x}} + \tilde{\mathbf{x}}^T \Delta S \tilde{\mathbf{x}} \quad (3.29)$$

for all ΔS, then K^* is truly the gain that minimizes the cost function. From Equation 3.27,

$$-\left(\dot{S}^*+\Delta\dot{S}\right)=\left(S^*+\Delta S\right)\left[A-B(K^*+\Delta K)\right]+\left[A-B(K^*+\Delta K)\right]^T\left(S^*+\Delta S\right) \\ +W+(K^*+\Delta K)^T V(K^*+\Delta K) \tag{3.30}$$

Subtracting Equation 3.27 from Equation 3.30 gives

$$-\Delta\dot{S}=\Delta S[A-B(K^*+\Delta K)]+[A-B(K^*+\Delta K)]^T\Delta S \\ +\Delta K^T V\Delta K+(K^{*T}V-S^*B)\Delta K+\Delta K^T(VK^*-B^TS^*) \tag{3.31}$$

Equation 3.31 is of the form of Equation 3.27, so the solution is

$$\Delta S(t,t_f)=\int_t^{t_f}\Phi_{cl}^T(\tau,t)[\Delta K^T V\Delta K+(K^{*T}V-S^*B)\Delta K \\ +\Delta K^T(VK^*-B^TS^*)]\Phi_{cl}(\tau,t)d\tau \tag{3.32}$$

where the time indices on S^*, K^*, and ΔK are omitted. Since ΔS must be positive semi-definite, then the non-quadratic terms,

$$(K^{*T}V-S^*B)\Delta K \tag{3.33}$$

and

$$\Delta K^T(VK^*-B^TS^*), \tag{3.34}$$

must be zero. Setting

$$K^{*T}V-S^*B=VK^*-B^TS^*=0, \tag{3.35}$$

then

$$K^*=V^{-1}B^TS^*. \tag{3.36}$$

Hence, the optimal gain is computed,

$$K(t)=V^{-1}B^TS(t,t_f). \tag{3.37}$$

The linear-quadratic-regulator is defined as the state-feedback control law of Equation 3.13 with the optimal gain of Equation 3.37. By back substituting Equation 3.37 into Equation 3.27,

$$-\dot{S}(t,t_f) = S(t,t_f)A + A^T S(t,t_f) - S(t,t_f)BV^{-1}B^T S(t,t_f) + W\,. \tag{3.38}$$

This equation is a Riccati equation and has the end condition given by Equation 3.28.

3.4 The Linear-Quadratic-Regulator

So far, the continuous controller depends on knowledge of a final time. Often, the final time is unknown or irrelevant. The Riccati equation has an asymptotic solution when the system is linear time-invariant, but since it is solved backwards in time, the "unchanging" part of the solution occurs near t_0. In other words, S varies mostly around t_f. Therefore, with an infinite horizon assumption,

$$\lim_{t_f \to \infty} \dot{S}(t,t_f) = 0. \tag{3.39}$$

In the limit, the Riccati equation becomes

$$0 = SA + A^T S - SBV^{-1}B^T S + W\,. \tag{3.40}$$

This equation is known as the algebraic Riccati equation, the solution to which is now constant. Only the positive definite solution will be considered. From Equation 3.36, it follows that

$$K = V^{-1}B^T S\,, \tag{3.41}$$

where K is also constant. The constant-coefficient control law

$$\mathbf{u}(t) = -K\tilde{\mathbf{x}}(t) \tag{3.42}$$

is known as a continuous time-invariant linear-quadratic regulator. Once the control gain, K, is constant, then the solution to the closed-loop differential equation becomes

$$\tilde{\mathbf{x}}(t) = \Phi_{cl}(t - t_0)\tilde{\mathbf{x}}(t_0) = e^{[A-BK](t-t_0)}\tilde{\mathbf{x}}(t_0). \tag{3.43}$$

3.5 Stability of the Linear-Quadratic-Regulator

Stengel[22] proves that the optimal control is always stable. First, define a positive definite Lyapunov function

$$\Im(\tilde{\mathbf{x}}, t) = \tilde{\mathbf{x}}^T(t) S \tilde{\mathbf{x}}(t), \tag{3.44}$$

where S is the positive definite solution to the algebraic Riccati equation. Stengel[22] shows that for asymptotic stability, the time derivative of the Lyapunov function must be negative definite at all times. Taking its derivative,

$$\dot{\Im}(\tilde{\mathbf{x}}, t) = \frac{\partial \Im}{\partial \tilde{\mathbf{x}}}(\tilde{\mathbf{x}}, t)\dot{\tilde{\mathbf{x}}}(t) = 2\tilde{\mathbf{x}}^T(t) S\left[A\tilde{\mathbf{x}}(t) + B\mathbf{u}(t)\right]. \tag{3.45}$$

Substituting Equations 3.41 and 3.42 for **u**, and reducing,

$$\begin{aligned} \dot{\Im}(\tilde{\mathbf{x}}, t) &= 2\tilde{\mathbf{x}}^T(t) S\left[A - BV^{-1}B^T S\right]\tilde{\mathbf{x}}(t) \\ &= \tilde{\mathbf{x}}^T(t)\left\{S\left[A - BV^{-1}B^T S\right] + \left[A - BV^{-1}B^T S\right]^T S\right\}\tilde{\mathbf{x}}(t). \\ &= \tilde{\mathbf{x}}^T(t)\left\{SA + A^T S - 2SBV^{-1}B^T S\right\}\tilde{\mathbf{x}}(t) \end{aligned} \tag{3.46}$$

From the algebraic Riccati equation (Equation 3.40),

$$SA + A^T S - 2SBV^{-1}B^T S = -W - SBV^{-1}B^T S; \tag{3.47}$$

thus,

$$\dot{\Im}(\tilde{\mathbf{x}}, t) = -\tilde{\mathbf{x}}^T(t)\left[W + SBV^{-1}B^T S\right]\tilde{\mathbf{x}}(t). \tag{3.48}$$

Because W is positive semi-definite, S is positive definite, and V is positive definite, then the time derivative of the Lyapunov function is negative for all times. Hence, the steady-state optimal control gain is stabilizing.

4. DISCRETE LINEAR QUADRATIC REGULATOR

4.1 Sampling a Continuous System

Often, a system is continuous, but only samples of the states and control are used. Usually, this is the case with spacecraft. Although the spacecraft move continuously, its position and velocity are measured in discrete intervals and computed using digital computers. Some thrusters can be operated continuously, but most are only fired intermittently. For most discrete systems, the sampling interval, or time step, is held constant. Now, we wish to find some similar corresponding discrete system,

$$\tilde{\mathbf{x}}_{k+1} = A_d \tilde{\mathbf{x}}_k + B_d \mathbf{u}_k, \tag{4.1}$$

to our known continuous system

$$\dot{\tilde{\mathbf{x}}}(t) = A\tilde{\mathbf{x}}(t) + B\mathbf{u}(t). \tag{4.2}$$

The subscript k, refers to the vector at time kT, where T is the sampling interval. In other words, for any vector $\mathbf{v}$,

$$\mathbf{v}_{k+m} = \mathbf{v}(kT + mT). \tag{4.3}$$

Also, the control vector, $\mathbf{u}$, is generic and not necessarily of the form

$$\mathbf{u}(t) = -K\tilde{\mathbf{x}}(t). \tag{3.42}$$

The solution to the differential equation with generic $\mathbf{u}$ is

$$\tilde{\mathbf{x}}(t) = \Phi(t - t_0)\tilde{\mathbf{x}}(t_0) + \int_{t_0}^{t} \Phi(t - \tau)B\mathbf{u}(\tau)d\tau. \tag{4.4}$$

Evaluating over one time interval,

$$t = kT + T \tag{4.5}$$

and

$$t_0 = kT\,. \tag{4.6}$$

Then,

$$\tilde{\mathbf{x}}(kT+T) = \Phi(T)\tilde{\mathbf{x}}(kT) + \int_{kT}^{kT+T} \Phi(kT+T-\tau)B\mathbf{u}(\tau)d\tau\,. \tag{4.7}$$

If the control vector has the zero order hold property (piecewise-constant), then

$$\mathbf{u}(t) = \mathbf{u}(kT) \tag{4.8}$$

for

$$kT \le t < kT+T\,, \tag{4.9}$$

and is constant over the period of integration. Thus,

$$\tilde{\mathbf{x}}_{k+1} = \Phi(T)\tilde{\mathbf{x}}_k + \left[\int_{kT}^{kT+T} \Phi(kT+T-\tau)Bd\tau\right]\mathbf{u}_k\,. \tag{4.10}$$

Phillips and Nagle[23] use a change of variable to simplify the integral, let

$$\sigma = kT+T-\tau\,. \tag{4.11}$$

Then the integral becomes

$$-\int_{T}^{0} \Phi(\sigma)Bd\sigma = \int_{0}^{T} \Phi(\sigma)Bd\sigma = \int_{0}^{T} \Phi(\tau)Bd\tau\,. \tag{4.12}$$

Therefore,

$$\tilde{\mathbf{x}}_{k+1} = \Phi(T)\tilde{\mathbf{x}}_k + \left[\int_{0}^{T} \Phi(\tau)Bd\tau\right]\mathbf{u}_k \tag{4.13}$$

has the desired form of Equation 4.1, where

$$A_d = \Phi(T) \tag{4.14}$$

and

$$B_d = \int_{0}^{T} \Phi(\tau)Bd\tau\,. \tag{4.15}$$

4.2 Discrete Optimal Control

Next, the continuous cost function is represented as a summation of integrals, rather than a single integral,

$$J = \sum_{k=0}^{k_f-1} \int_{t_k}^{t_{k+1}} \left\{ \tilde{\mathbf{x}}^T(t) W \tilde{\mathbf{x}}(t) + \mathbf{u}^T(t) V \mathbf{u}(t) \right\} dt , \tag{4.16}$$

where k_f corresponds to the final time, t_f. Assuming piecewise-constant control from above, then over the period of integration

$$\tilde{\mathbf{x}}(t) = A_d \tilde{\mathbf{x}}_k + B_d \mathbf{u}_k , \tag{4.17}$$

and

$$\mathbf{u}(t) = \mathbf{u}_k . \tag{4.18}$$

Substituting Equations 4.17 and 4.18 into the cost function, Equation 4.16, and reducing,

$$\begin{aligned} J &= \sum_{k=0}^{k_f-1} \int_{t_k}^{t_{k+1}} \left\{ (\tilde{\mathbf{x}}_k^T A_d^T + \mathbf{u}_k^T B_d^T) W (A_d \tilde{\mathbf{x}}_k + B_d \mathbf{u}_k) + \mathbf{u}_k^T V \mathbf{u}_k \right\} dt \\ &= \sum_{k=0}^{k_f-1} \int_{t_k}^{t_{k+1}} \{ \tilde{\mathbf{x}}_k^T A_d^T W A_d \tilde{\mathbf{x}}_k + \tilde{\mathbf{x}}_k^T A_d^T W B_d \mathbf{u}_k + \mathbf{u}_k^T B_d^T W A_d \tilde{\mathbf{x}}_k \\ &\qquad + \mathbf{u}_k^T B_d^T W B_d \mathbf{u}_k + \mathbf{u}_k^T V \mathbf{u}_k \} dt \\ &= \sum_{k=0}^{k_f-1} \int_{t_k}^{t_{k+1}} \left\{ \begin{bmatrix} \tilde{\mathbf{x}}_k^T & \mathbf{u}_k^T \end{bmatrix} \begin{bmatrix} A_d^T W A_d & A_d^T W B_d \\ B_d^T W A_d & (B_d^T W B_d + V) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_k \\ \mathbf{u}_k \end{bmatrix} \right\} dt \end{aligned} \tag{4.19}$$

Both $\tilde{\mathbf{x}}_k$ and $\mathbf{u}_k$ are constant over the period of integration and can be removed from the integral

$$J = \sum_{k=0}^{k_f-1} \begin{bmatrix} \tilde{\mathbf{x}}_k^T & \mathbf{u}_k^T \end{bmatrix} \int_{t_k}^{t_{k+1}} \begin{bmatrix} A_d^T W A_d & A_d^T W B_d \\ B_d^T W A_d & (B_d^T W B_d + V) \end{bmatrix} dt \begin{bmatrix} \tilde{\mathbf{x}}_k \\ \mathbf{u}_k \end{bmatrix} . \tag{4.20}$$

This can be rewritten as

$$J = \sum_{k=0}^{k_f-1} \begin{bmatrix} \tilde{\mathbf{x}}_k^T & \mathbf{u}_k^T \end{bmatrix} \begin{bmatrix} W_d & M_d \\ M_d^T & V_d \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_k \\ \mathbf{u}_k \end{bmatrix}, \tag{4.21}$$

where

$$W_d = \int_{t_k}^{t_{k+1}} A_d^T W A_d dt, \tag{4.22}$$

$$V_d = \int_{t_k}^{t_{k+1}} (B_d^T W B_d + V) dt, \tag{4.23}$$

and

$$M_d = \int_{t_k}^{t_{k+1}} A_d^T W B_d dt. \tag{4.24}$$

These integrals remain constant as long as the sampling interval is constant; therefore, they can be expressed as

$$W_d = \int_0^T A_d^T W A_d dt, \tag{4.25}$$

$$V_d = \int_0^T (B_d^T W B_d + V) dt, \tag{4.26}$$

and

$$M_d = \int_0^T A_d^T W B_d dt. \tag{4.27}$$

Special care must be given to the imbedded integral, B_d. The upper limit must be altered from the time step, to the variable of integration for the outer integral when calculating V_d and M_d. Thus,

$$B_d = \int_0^T \Phi(\tau) B d\tau = \int_0^t \Phi(\tau) B d\tau. \tag{4.28}$$

The coupled state-control weighting term, M_d, is interesting. In the continuous system, no coupling exists (although linear quadratic regulator theory could be modified to accommodate it). However, once sampling occurs and the system becomes discrete, a coupling term is introduced.

Next, the cost function is modified to include a weighting on the state error at the final time. Also, a factor of one-half is added to simplify the math. This has no effect because optimizing J would be the same as optimizing $2J$.

$$J = \frac{1}{2}\tilde{\mathbf{x}}_{k_f}^T S_{k_f} \tilde{\mathbf{x}}_{k_f} + \sum_{k=0}^{k_f-1} \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{x}}_k^T & \mathbf{u}_k^T\end{bmatrix}\begin{bmatrix}W_d & M_d \\ M_d^T & V_d\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_k \\ \mathbf{u}_k\end{bmatrix}. \tag{4.29}$$

Eventually, this final state error will go to zero as the final time goes to infinity, but is necessary for the development of the theory. Next, a Lagrange multiplier is appended to the cost function inside the summation

$$J = \frac{1}{2}\tilde{\mathbf{x}}_{k_f}^T S_{k_f} \tilde{\mathbf{x}}_{k_f} + \sum_{k=0}^{k_f-1} \{\frac{1}{2}\begin{bmatrix}\tilde{\mathbf{x}}_k^T & \mathbf{u}_k^T\end{bmatrix}\begin{bmatrix}W_d & M_d \\ M_d^T & V_d\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_k \\ \mathbf{u}_k\end{bmatrix}. \tag{4.30}$$
$$+\lambda_{k+1}^T(A_d\tilde{\mathbf{x}}_k + B_d\mathbf{u}_k - \tilde{\mathbf{x}}_{k+1})\}$$

From Equation 4.1, the last term in Equation 4.30 is zero, so the value of the cost function remains unchanged. The Lagrange multiplier, λ_k, is also called the co-state vector. The cost function is rewritten as

$$J = \frac{1}{2}\tilde{\mathbf{x}}_{k_f}^T S_{k_f} \tilde{\mathbf{x}}_{k_f} + \sum_{k=0}^{k_f-1} \{H_k - \lambda_{k+1}^T \tilde{\mathbf{x}}_{k+1}\}, \tag{4.31}$$

where

$$H_k = \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{x}}_k^T & \mathbf{u}_k^T\end{bmatrix}\begin{bmatrix}W_d & M_d \\ M_d^T & V_d\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_k \\ \mathbf{u}_k\end{bmatrix} + \lambda_{k+1}^T(A_d\tilde{\mathbf{x}}_k + B_d\mathbf{u}_k), \tag{4.32}$$

and H_k is referred to as the Hamiltonian. Changing the lower limit of summation from 0 to 1,

$$J = \frac{1}{2}\tilde{\mathbf{x}}_{k_f}^T S_{k_f} \tilde{\mathbf{x}}_{k_f} + H_0 - \lambda_{k_f}^T \tilde{\mathbf{x}}_{k_f} + \sum_{k=1}^{k_f-1} \{H_k - \lambda_k^T \tilde{\mathbf{x}}_k\}. \tag{4.33}$$

Assume a function, $f(x_1, x_2, \ldots, x_n)$, exists with a minimum, $f_{\min}(x_1^*, x_2^*, \ldots, x_n^*)$. A location near the minimum can be written as

$$f(x_1, x_2, \ldots, x_n) = f_{\min}(x_1^*, x_2^*, \ldots, x_n^*) + \left.\frac{\partial f}{\partial x_1}\right|_* \delta x_1 + \left.\frac{\partial f}{\partial x_2}\right|_* \delta x_2 + \cdots + \left.\frac{\partial f}{\partial x_n}\right|_* \delta x_n + (\textit{higher order terms}) \tag{4.34}$$

The first variation of f is defined as

$$\delta f = f(x_1, x_2, \ldots, x_n) - f_{\min}(x_1^*, x_2^*, \ldots, x_n^*) = \left.\frac{\partial f}{\partial x_1}\right|_* \delta x_1 + \left.\frac{\partial f}{\partial x_2}\right|_* \delta x_2 + \cdots + \left.\frac{\partial f}{\partial x_n}\right|_* \delta x_n + (\textit{higher order terms}) \tag{4.35}$$

Assuming the variation is very small, or f is very near $f_{\min}$, the higher order terms can be neglected. Reference Bryson and Ho[24] or Stengel[22] for more information on the topic of optimality. Furthermore, the following derivation for discrete optimal control is standard and used in both Bryson and Ho[24] and Stengel[22]. Additionally, this technique could be used to develop continuous optimal control rather than the Friedland[21] method used in Section 3.3.

Setting the first variation of the cost function (Equation 4.33) to zero will provide necessary conditions for optimality. The first variation is

$$\delta J = \left[\frac{\partial \frac{1}{2} \tilde{\mathbf{x}}_{k_f}^T S_{k_f} \tilde{\mathbf{x}}_{k_f}}{\partial \tilde{\mathbf{x}}_{k_f}} - \frac{\partial \lambda_{k_f}^T \tilde{\mathbf{x}}_{k_f}}{\partial \tilde{\mathbf{x}}_{k_f}} \right] \delta \tilde{\mathbf{x}}_{k_f} + \frac{\partial H_0}{\partial \tilde{\mathbf{x}}_0} \delta \tilde{\mathbf{x}}_0 + \frac{\partial H_0}{\partial \mathbf{u}_0} \delta \mathbf{u}_0 + \sum_{k=1}^{k_f-1} \left\{ \left[\frac{\partial H_k}{\partial \tilde{\mathbf{x}}_k} - \frac{\partial \lambda_k^T \tilde{\mathbf{x}}_k}{\partial \tilde{\mathbf{x}}_k} \right] \delta \tilde{\mathbf{x}}_k + \frac{\partial H_k}{\partial \mathbf{u}_k} \delta \mathbf{u}_k \right\} \quad (4.36)$$

The variation with respect to the co-state has been omitted for brevity. If it were included, it would simply lead back to the original difference equation constraint. To ensure that δJ is zero, all individual variations within must be zero. The initial state error vector and initial control vector are constant, so their variations are inherently zero

$$\delta \tilde{\mathbf{x}}_0 = 0 \quad (4.37)$$

and

$$\delta \mathbf{u}_0 = 0. \quad (4.38)$$

Evaluating the partials in Equation 4.36 and setting them to zero leaves the three vector equations,

$$\tilde{\mathbf{x}}_{k_f}^T S_{k_f} = \lambda_{k_f}^T, \quad (4.39)$$

$$\lambda_k^T = \frac{\partial H_k}{\partial \tilde{\mathbf{x}}_k} = \tilde{\mathbf{x}}_k^T W_d + \mathbf{u}_k^T M_d^T + \lambda_{k+1}^T A_d, \quad (4.40)$$

and

$$0 = \frac{\partial H_k}{\partial \mathbf{u}_k} = \tilde{\mathbf{x}}_k^T M_d + \mathbf{u}_k^T V_d + \lambda_{k+1}^T B_d. \quad (4.41)$$

These equations are known as the Euler-Lagrange equations. Because the final time is arbitrary, it stands from Equation 4.39 that

$$\lambda_k^T = \tilde{\mathbf{x}}_k^T S_k, \quad (4.42)$$

and

$$\lambda_{k+1} = S_{k+1}\tilde{\mathbf{x}}_{k+1}, \tag{4.43}$$

for all sampling instances. Substituting Equation 4.1 into Equation 4.43,

$$\lambda_{k+1} = S_{k+1}(A_d\tilde{\mathbf{x}}_k + B_d\mathbf{u}_k), \tag{4.44}$$

and substituting Equation 4.44 into Equation 4.41,

$$\begin{aligned} 0 &= \tilde{\mathbf{x}}_k^T M_d + \mathbf{u}_k^T V_d + \tilde{\mathbf{x}}_k^T A_d^T S_{k+1} B_d + \mathbf{u}_k^T B_d^T S_{k+1} B_d \\ &= \tilde{\mathbf{x}}_k^T (M_d + A_d^T S_{k+1} B_d) + \mathbf{u}_k^T (V_d + B_d^T S_{k+1} B_d) \end{aligned} \tag{4.45}$$

Taking the transpose,

$$0 = (M_d^T + B_d^T S_{k+1} A_d)\tilde{\mathbf{x}}_k + (V_d + B_d^T S_{k+1} B_d)\mathbf{u}_k. \tag{4.46}$$

Note that W, V, and S_k, and in turn W_d and V_d, are at least positive semi-definite, so the control can be found from

$$\mathbf{u}_k = -(V_d + B_d^T S_{k+1} B_d)^{-1}(M_d^T + B_d^T S_{k+1} A_d)\tilde{\mathbf{x}}_k, \tag{4.47}$$

which is in state-feedback form. The control law is then

$$\mathbf{u}_k = -K_d\tilde{\mathbf{x}}_k, \tag{4.48}$$

where the discrete control gain matrix is

$$K_d = (V_d + B_d^T S_{k+1} B_d)^{-1}(M_d^T + B_d^T S_{k+1} A_d). \tag{4.49}$$

Substituting the transpose of Equation 4.44 into Equation 4.40 gives

$$\lambda_k^T = \tilde{\mathbf{x}}_k^T W_d + \mathbf{u}_k^T M_d^T + (\tilde{\mathbf{x}}_k^T A_d^T + \mathbf{u}_k^T B_d^T) S_{k+1} A_d, \tag{4.50}$$

and transposing,

$$\lambda_k = W_d\tilde{\mathbf{x}}_k + M_d\mathbf{u}_k + A_d^T S_{k+1} A_d\tilde{\mathbf{x}} + A_d^T S_{k+1} B_d\mathbf{u}_k. \tag{4.51}$$

Substituting Equation 4.42 for the co-state and combining like terms,

$$S_k\tilde{\mathbf{x}}_k = (W_d + A_d^T S_{k+1} A_d)\tilde{\mathbf{x}}_k + (M_d + A_d^T S_{k+1} B_d)\mathbf{u}_k. \tag{4.52}$$

Replacing the control with its state-feedback form and dropping $\tilde{\mathbf{x}}_k$ from all terms leaves the discrete Riccati equation:

$$S_k = W_d + A_d^T S_{k+1} A_d - (M_d^T + B_d^T S_{k+1} A_d)^T (V_d + B_d^T S_{k+1} B_d)^{-1} (M_d^T + B_d^T S_{k+1} A_d) \quad (4.53)$$

This equation is solved backward in time with the known end condition, S_{k_f}. However, just as with the continuous case, S reaches a steady-state value at the beginning. The transient part of S occurs near the final time. If the final time goes to infinity, then S is considered constant. Then,

$$\lim_{k \to \infty} S_k = S_{k+1} = S, \quad (4.54)$$

and the discrete algebraic Riccati equation becomes

$$0 = W_d + A_d^T S A_d - S - (M_d^T + B_d^T S A_d)^T (V_d + B_d^T S B_d)^{-1} (M_d^T + B_d^T S A_d). \quad (4.55)$$

Furthermore, the discrete control gain matrix is now constant

$$K_d = (V_d + B_d^T S B_d)^{-1} (M_d^T + B_d^T S A_d). \quad (4.56)$$

A continuous optimal control method has been found. Once the system is sampled, with a sampling interval the same as the maneuver interval, the continuous system is transformed into a discrete system. Finally, an optimal discrete control method has been built.

5. FORMATION FLYING

5.1 Uncoupled Satellite Control

So far, all the development has focused on the control of only one satellite. For many future missions, multiple satellites will be required. Adding additional satellites to the state-space form is done by making the matrices block diagonal and appending additional satellite states on the state error vector. For the continuous case example from Equation 4.2, the state error vector becomes

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_1 - \mathbf{x}_{ref1} \\ \mathbf{x}_2 - \mathbf{x}_{ref2} \\ \vdots \\ \mathbf{x}_j - \mathbf{x}_{refj} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \vdots \\ \tilde{\mathbf{x}}_j \end{bmatrix}, \tag{5.1}$$

the control vector becomes

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_j \end{bmatrix}^T, \tag{5.2}$$

the plant dynamics matrix becomes

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_j \end{bmatrix}, \tag{5.3}$$

and the control-mapping matrix becomes

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_j \end{bmatrix}. \tag{5.4}$$

The numerical subscript refers to the satellite number, with subscript j being the last satellite included in the single condensed equation.

The discrete system is appended similarly. The difference equation from Equation 4.1 remains, but

$$\tilde{\mathbf{x}}_k = \begin{bmatrix} \tilde{\mathbf{x}}_{k1} & \tilde{\mathbf{x}}_{k2} & \cdots & \tilde{\mathbf{x}}_{kj} \end{bmatrix}^T, \tag{5.5}$$

$$\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{k1} & \mathbf{u}_{k2} & \cdots & \mathbf{u}_{kj} \end{bmatrix}^T, \tag{5.6}$$

$$A_d = \begin{bmatrix} A_{d1} & & & \\ & A_{d2} & & \\ & & \ddots & \\ & & & A_{dj} \end{bmatrix}, \tag{5.7}$$

and

$$B_d = \begin{bmatrix} B_{d1} & & & \\ & B_{d2} & & \\ & & \ddots & \\ & & & B_{dj} \end{bmatrix}. \tag{5.8}$$

Satellites in close proximity to each other usually have the same plant dynamics sub-matrix and control-mapping sub-matrix. Therefore,

$$A_1 = A_i = A_j \tag{5.9}$$

$$A_{d1} = A_{di} = A_{dj} \tag{5.10}$$

$$B_1 = B_i = B_j \tag{5.11}$$

$$B_{d1} = B_{di} = B_{dj}, \tag{5.12}$$

for all i and j. Appending the system in this manner allows for multiple satellites, but they remain uncoupled.

5.2 Coupled Satellite Control

Formation flying requires that at least one of the satellites is controlled relative to another satellite. With the uncoupled system, there is no relative control. Wagner[25] shows that to control one satellite relative to another the differential equations of the two satellites are simply subtracted. For example, if satellite 2 is controlled relative to satellite 1, then

$$\dot{\mathbf{x}}_2 - \dot{\mathbf{x}}_1 = A(\mathbf{x}_2 - \mathbf{x}_1) + B(\mathbf{u}_2 - \mathbf{u}_1). \tag{5.13}$$

This is rewritten as

$$\dot{\mathbf{x}}_{rel} = A\mathbf{x}_{rel} + B\mathbf{u}_2 - B\mathbf{u}_1, \tag{5.14}$$

where $\mathbf{x}_{rel}$ is the relative state vector. By superposition,

$$\dot{\tilde{\mathbf{x}}}_{rel} = A\tilde{\mathbf{x}}_{rel} + B\mathbf{u}_2 - B\mathbf{u}_1. \tag{5.15}$$

If the state error vector of satellite 2 is then redefined as the relative state error vector compared to satellite 1, then

$$\dot{\tilde{\mathbf{x}}}_2 = A\tilde{\mathbf{x}}_2 + B\mathbf{u}_2 - B\mathbf{u}_1. \tag{5.16}$$

Similarly for the discrete system,

$$\tilde{\mathbf{x}}_{(k+1)2} = A_d\tilde{\mathbf{x}}_{k2} + B_d\mathbf{u}_{k2} - B_d\mathbf{u}_{k1}. \tag{5.17}$$

If all additional satellites are controlled relative to the first satellite, then the continuous state-space system formed from Equations 4.2, 5.1, 5.2, and 5.3 holds, but the control-mapping matrix becomes

$$B = \begin{bmatrix} B_1 & & & & \\ -B_1 & B_i & & & \\ -B_1 & & B_i & & \\ \vdots & & & \ddots & \\ -B_1 & & & & B_j \end{bmatrix}. \tag{5.19}$$

For the discrete system, Equations 4.2, 5.5, 5.6, and 5.7 remain the same, but the discrete control-mapping matrix changes to

$$B_d = \begin{bmatrix} B_{d1} & & & & \\ -B_{d1} & B_{di} & & & \\ -B_{d1} & & B_{di} & & \\ \vdots & & & \ddots & \\ -B_{d1} & & & & B_{dj} \end{bmatrix}. \tag{5.20}$$

For both the continuous system and discrete system, the additional "drone" satellites are now controlled relative to the first "hub" satellite. The reference trajectory of the hub satellite is whatever stellar trajectory is desired. A Lissajous orbit around the sun-Earth L2 point is an example. A reference trajectory of a drone satellite is whatever relative motion or position is desired about the hub satellite. Positioning a drone satellite at a point one kilometer in the negative along-track direction, with no relative velocity, would have the desired reference trajectory

$$\mathbf{x}_{ref2} = [0 \quad -1 \quad 0 \quad 0 \quad 0 \quad 0]^T. \tag{5.21}$$

6. DISCRETE KALMAN FILTER

6.1 Estimates and Measurements

Thus far, the system assumes perfect mathematical modeling and no noises or disturbances. In real life, however, systems are not ideal. Also, the true state (or state error) is directly implemented in the control scheme. Often, the truth is unknown and an estimate of the state (or state error) must be used. A Kalman filter blends together state estimates and noisy sensor measurements of the true states. This is a method of reducing noise and providing the best estimate of the actual unknown states (or state errors).

Process noise is added to the discrete system

$$\tilde{\mathbf{x}}_{k+1} = A_d \tilde{\mathbf{x}}_k + B_d \mathbf{u}_k + w, \tag{6.1}$$

where w is the random process noise vector. The process noise vector is assumed to be Gaussian white noise (zero-mean, $E[w] = 0$) with a known covariance structure

$$E\left[ww^T\right] = Q, \tag{6.2}$$

where $E[a]$ is the expected value of a as defined in Brown and Hwang[26]. The process noise is included to account for unmodeled disturbances, plant errors, and plant nonlinearities. The process noise covariance matrix, Q, is diagonal and symmetric.

The measurements are expressed as a linear combination of the states (or state errors):

$$\mathbf{y}_k = C\tilde{\mathbf{x}}_k + v, \tag{6.3}$$

where v is the measurement noise vector. If the satellite's Cartesian position coordinates are measured exactly, then

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{6.4}$$

Similar to the process noise, the measurement noise is also assumed to be Gaussian white noise with a known covariance structure

$$E\left[vv^T\right] = R. \tag{6.5}$$

The measurement noise covariance matrix, R, is also diagonal, invertible, and symmetric. Measurement noise, or sensor noise, is included to account for the fact that sensors are not ideal. For this thesis, there is no correlation between v and w. Eventually, the Kalman filter will be modified to handle measurements that are nonlinear functions of the state error.

6.2 Multiple Satellite Measurements

For multiple satellites, the measurement equation is appended such that

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{y}_{k1} \\ \mathbf{y}_{k2} \\ \vdots \\ \mathbf{y}_{kj} \end{bmatrix}, \tag{6.6}$$

and

$$C = \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & \ddots & \\ & & & C_j \end{bmatrix}. \tag{6.7}$$

If the type of measurements are the same for each satellite, then C_i is the same for all i. There is also an implicit assumption that measurements for one satellite are uncoupled

from measurements for another. Similar to the state errors, however, the measurements for the drone satellites are relative to the hub satellite. The hub satellite measurements are relative to the origin of the coordinate system in use. If the circular restricted three-body dynamics are in place, then the measurements would be relative to the libration point.

6.3 Optimal Estimate Updating

As mentioned before, the true state error is not known and a state error estimate must be used. The estimate is expressed in two ways. The *a priori* estimate, $\hat{\mathbf{x}}_k^-$, represents the estimate prior to assimilating the measurements at time t_k. The *a posteriori* estimate, $\hat{\mathbf{x}}_k$, represents the estimate after assimilating the measurements. The *a posteriori* (updated) estimate is propagated with the same dynamic model as the actual state error, except noise is excluded since it has zero-mean.

$$\hat{\mathbf{x}}_{k+1}^- = A_d \hat{\mathbf{x}}_k + B_d \mathbf{u}_k . \tag{6.8}$$

Propagating the updated estimate gives the *a priori* estimate at the next time step.

The optimal blending of the measurement with the *a priori* estimate to find the updated estimate can be found by minimizing another cost function,

$$J_e = \frac{1}{2}\left[(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)^T P_k^{-1}(-)(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-) + (\mathbf{y}_k - C\hat{\mathbf{x}}_k)^T R^{-1}(\mathbf{y}_k - C\hat{\mathbf{x}}_k)\right], \tag{6.9}$$

where

$$P_k(-) = E[(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)^T], \tag{6.10}$$

is the covariance matrix of the estimation error before measurements are included, and later,

$$P_k(+) = E[(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)^T], \tag{6.11}$$

is the covariance matrix of the estimation error after measurements are included. Minimizing this cost function has two goals. The first is to force the difference between the updated estimate and the *a priori* estimate to be small, meaning the updated estimate is close to the old estimate. The second is to force the difference between the actual measurement and the expected measurement to be small. If both these conditions are minimal, then the estimated state error would be very near the actual state error. If the cost function is found to be zero, then the estimate is perfect. Taking the derivative of the cost function with respect to the updated estimate and setting it to zero gives

$$\frac{\partial J_e}{\partial \hat{\mathbf{x}}_k} = \left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-\right)^T P_k^{-1}(-) - \left(\mathbf{y}_k - C\hat{\mathbf{x}}_k\right)^T R^{-1} C = 0. \tag{6.12}$$

Next, transpose the equation,

$$0 = P_k^{-1}(-)\left(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-\right) - C^T R^{-1}\left(\mathbf{y}_k - C\hat{\mathbf{x}}_k\right), \tag{6.13}$$

and move the updated estimate term to the other side,

$$\left(P_k^{-1}(-) + C^T R^{-1} C\right)\hat{\mathbf{x}}_k = P_k^{-1}(-)\hat{\mathbf{x}}_k^- + C^T R^{-1}\mathbf{y}_k. \tag{6.14}$$

Solving for the updated estimate,

$$\hat{\mathbf{x}}_k = \left(P_k^{-1}(-) + C^T R^{-1} C\right)^{-1}\left(P_k^{-1}(-)\hat{\mathbf{x}}_k^- + C^T R^{-1}\mathbf{y}_k\right), \tag{6.15}$$

and adding and subtracting the term $C^T R^{-1} C\hat{\mathbf{x}}_k^-$ from the equation,

$$\hat{\mathbf{x}}_k = \left(P_k^{-1}(-) + C^T R^{-1} C\right)^{-1}\left(P_k^{-1}(-)\hat{\mathbf{x}}_k^- + C^T R^{-1} C\hat{\mathbf{x}}_k^- - C^T R^{-1} C\hat{\mathbf{x}}_k^- + C^T R^{-1}\mathbf{y}_k\right). \tag{6.16}$$

This reduces to

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + L_k\left(\mathbf{y}_k - C\hat{\mathbf{x}}_k^-\right), \tag{6.17}$$

where

$$L_k = \left(P_k^{-1}(-) + C^T R^{-1} C\right)^{-1} C^T R^{-1} \tag{6.18}$$

is the estimate gain, filter gain, or Kalman gain. The updated estimate is now expressed as the optimal linear combination of the *a priori* estimate and measurements.

6.4 The Kalman Gain

If n is the number of states and m is the number of measurements, then the matrix, $\left(P_k^{-1}(-) + C^T R^{-1} C\right)$, is size n by n. Usually, the number of measurements is less than the number of states ($m<n$). Therefore, it is computationally easier to invert a size m by m matrix than an n by n matrix. Rather than just simply inverting the n by n matrix, a matrix inversion lemma is used so the only inversion is done on matrices of size m by m. Let

$$F = \left(P_k^{-1}(-) + C^T R^{-1} C\right)^{-1}, \tag{6.19}$$

then,

$$P_k^{-1}(-) + C^T R^{-1} C = F^{-1}. \tag{6.20}$$

Pre-multiply both sides by F,

$$F P_k^{-1}(-) + F C^T R^{-1} C = I, \tag{6.21}$$

post-multiply both sides by $P_k(-)$,

$$F + F C^T R^{-1} C P_k(-) = P_k(-), \tag{6.22}$$

post-multiply both sides by C^T,

$$F C^T + F C^T R^{-1} C P_k(-) C^T = P_k(-) C^T, \tag{6.23}$$

and factor out to the left $F C^T R^{-1}$,

$$FC^T R^{-1}(R + CP_k(-)C^T) = P_k(-)C^T. \tag{6.24}$$

Post-multiply by the inverse of the m by m matrix,

$$FC^T R^{-1} = P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1}, \tag{6.25}$$

and post-multiply by $CP_k(-)$,

$$FC^T R^{-1}CP_k(-) = P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1}CP_k(-). \tag{6.26}$$

Subtracting Equation 6.26 from Equation 6.22 reveals

$$F = P_k(-) - P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1}CP_k(-), \tag{6.27}$$

or

$$\left(P_k^{-1}(-) + C^T R^{-1}C\right)^{-1} = P_k(-)\left\{I - C^T\left(R + CP_k(-)C^T\right)^{-1}CP_k(-)\right\}. \tag{6.28}$$

Substituting the result of the matrix inversion lemma, Equation 6.28, into the Kalman gain equation, Equation 6.18, gives

$$L_k = P_k(-)\left\{I - C^T\left(R + CP_k(-)C^T\right)^{-1}CP_k(-)\right\}C^T R^{-1}. \tag{6.29}$$

This is rewritten as

$$L_k = P_k(-)C^T\left\{I - \left(R + CP_k(-)C^T\right)^{-1}CP_k(-)C^T\right\}R^{-1}. \tag{6.30}$$

Factoring out the inverse term gives

$$L_k = P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1}\left\{\left(R + CP_k(-)C^T\right) - CP_k(-)C^T\right\}R^{-1}, \tag{6.31}$$

which reduces to

$$L_k = P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1}. \tag{6.32}$$

This is the common equation found in literature for the optimal Kalman gain.

6.5 Estimation Error Covariance Propagation

Now that the estimate can be updated and propagated, the estimation error covariance matrix must be updated and propagated as well. Going back to the definition of the *a priori* estimation error covariance matrix in Equation 6.10, it stands to reason that at the next time step

$$P_{k+1}(-) = E[(\tilde{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_{k+1}^-)(\tilde{\mathbf{x}}_{k+1} - \hat{\mathbf{x}}_{k+1}^-)^T]. \tag{6.33}$$

Substituting Equation 6.1 into Equation 6.33,

$$\begin{aligned} P_{k+1}(-) = E[&(A_d\tilde{\mathbf{x}}_k + B_d\mathbf{u}_k + w - A_d\hat{\mathbf{x}}_k - B_d\mathbf{u}_k) \\ &(A_d\tilde{\mathbf{x}}_k + B_d\mathbf{u}_k + w - A_d\hat{\mathbf{x}}_k - B_d\mathbf{u}_k)^T]' \end{aligned} \tag{6.34}$$

which reduces to

$$P_{k+1}(-) = E[(A_d(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k) + w)(A_d(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k) + w)^T]. \tag{6.35}$$

Expanding yields

$$\begin{aligned} P_{k+1}(-) = E[A_d(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)^T A_d^T] + E[A_d(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)w^T] + \\ E[w(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)^T A_d^T] + E[ww^T] \end{aligned}. \tag{6.36}$$

Assuming the process noise and estimation error have no correlation, then

$$E[A_d(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)w^T] = E[w(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)^T A_d^T] = 0. \tag{6.37}$$

Also, A_d has no randomness and can be removed from the expectation, so

$$P_{k+1}(-) = A_d E[(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k)^T]A_d^T + E[ww^T]. \tag{6.38}$$

Then, from previous definitions (Equations 6.2 and 6.11),

$$P_{k+1}(-) = A_d P_k(+)A_d^T + Q. \tag{6.39}$$

This is the propagation equation for the estimation error covariance matrix.

6.6 Estimation Error Covariance Updating

The definition of the updated estimation error covariance matrix is given in Equation 6.11. Substituting Equation 6.17 into Equation 6.11,

$$P_k(+) = E[(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^- - L_k(\mathbf{y}_k - C\hat{\mathbf{x}}_k^-))(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^- - L_k(\mathbf{y}_k - C\hat{\mathbf{x}}_k^-))^T]. \tag{6.40}$$

Next, substitute in Equation 6.3 and multiply through,

$$\begin{aligned} P_k(+) = E[&(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^- - L_k C\tilde{\mathbf{x}}_k - L_k v - L_k C\hat{\mathbf{x}}_k^-) \\ &(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^- - L_k C\tilde{\mathbf{x}}_k - L_k v - L_k C\hat{\mathbf{x}}_k^-)^T]. \end{aligned} \tag{6.41}$$

Combining like terms,

$$P_k(+) = E[((I - L_k C)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-) - L_k v)((I - L_k C)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-) - L_k v)^T], \tag{6.42}$$

and expanding,

$$\begin{aligned} P_k(+) = {} & E[(I - L_k C)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)^T (I - L_k C)^T] - \\ & E[(I - L_k C)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)v^T L_k^T] - E[L_k v(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)^T (I - L_k C)^T] + . \\ & E[L_k v v^T L_k^T] \end{aligned} \tag{6.43}$$

Assuming that the measurement noise has no correlation to the *a priori* estimation error, then

$$E[(I - L_k C)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)v^T L_k^T] = E[L_k v(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)^T (I - L_k C)^T] = 0. \tag{6.44}$$

Extracting from the expectations constant terms (over the time step) gives

$$P_k(+) = (I - L_k C)E[(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)(\tilde{\mathbf{x}}_k - \hat{\mathbf{x}}_k^-)^T](I - L_k C)^T + L_k E[vv^T]L_k^T. \tag{6.45}$$

Using previous definitions (Equations 6.10 and 6.5), this can be rewritten as

$$P_k(+) = (I - L_k C)P_k(-)(I - L_k C)^T + L_k R L_k^T, \tag{6.46}$$

which is known as the Joseph formula. Multiplying through,

$$P_k(+) = P_k(-) - P_k(-)C^T L_k^T - L_k C P_k(-) + L_k C P_k(-) C^T L_k^T + L_k R L_k^T, \tag{6.47}$$

and substituting the Kalman gain, Equation 6.32, gives

$$P_k(+) = P_k(-) - L_k C P_k(-) + P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1} CP_k(-)C^T L_k^T + P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1} RL_k^T - P_k(-)C^T L_k^T \quad (6.48)$$

Factoring $P_k(-)C^T$ to the left and L_k^T to the right yields

$$P_k(+) = P_k(-) - L_k CP_k(-) + P_k(-)C^T\left\{\left(R + CP_k(-)C^T\right)^{-1} CP_k(-)C^T + \left(R + CP_k(-)C^T\right)^{-1} R - I\right\}L_k^T \quad (6.49)$$

Further factoring inside the braces reveals

$$P_k(+) = P_k(-) - L_k CP_k(-) + P_k(-)C^T\left\{\left(R + CP_k(-)C^T\right)^{-1}\left(R + CP_k(-)C^T\right) - I\right\}L_k^T, \quad (6.50)$$

and

$$P_k(+) = P_k(-) - L_k CP_k(-) + P_k(-)C^T\{I - I\}L_k^T. \quad (6.51)$$

Thus,

$$P_k(+) = P_k(-) - L_k CP_k(-) = (I - L_k C)P_k(-) \quad (6.52)$$

is the update equation for the estimation error covariance matrix. The Joseph Equation, Equation 6.46, is numerically stable, as it preserves a positive definite $P_k(+)$ for any gain. Equation 6.52 is correct only for the exact optimal gain. Truncation and round-off errors may lead to a non-positive definite $P_k(+)$.

6.7 Kalman Filter Algorithm

Brown and Hwang[26] give the following algorithm for the Kalman filter:

1. Enter prior estimate, $\hat{\mathbf{x}}_k^-$, and its error covariance, $P_k(-)$.

2. Compute Kalman gain,

$$L_k = P_k(-)C^T\left(R + CP_k(-)C^T\right)^{-1}. \quad (6.32)$$

3. Update estimate with measurement,

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + L_k\left(\mathbf{y}_k - C\hat{\mathbf{x}}_k^-\right). \tag{6.17}$$

4. Compute error covariance for updated estimate,

$$P_k(+) = (I - L_k C)P_k(-). \tag{6.52}$$

Here, Equation 6.46 may be substituted for Equation 6.52 to avoid truncation and round-off errors.

5. Propagate estimate forward,

$$\hat{\mathbf{x}}_{k+1}^- = A_d\hat{\mathbf{x}}_k + B_d\mathbf{u}_k. \tag{6.8}$$

6. Propagate error covariance forward,

$$P_{k+1}(-) = A_d P_k(+)A_d^T + Q. \tag{6.39}$$

7. Repeat.

6.8 Adjusting for Continuous System Noise

In Equation 6.1, noise is appended to the discrete system. If noise is appended to the continuous system,

$$\dot{\tilde{\mathbf{x}}}(t) = A\tilde{\mathbf{x}}(t) + B\mathbf{u}(t) + w_c(t), \tag{6.53}$$

where the Gaussian white noise strength is

$$E[w_c(t)w_c^T(\tau)] = Q_c(t)\delta(t-\tau), \tag{6.54}$$

then it must be discretized to match the form in Equation 6.1. From Section 4.1, we know the discrete process noise vector, w, is determined by

$$w = \int_0^T \Phi(\tau)w_c(\tau)d\tau. \tag{6.55}$$

Substituting Equation 6.55 into Equation 6.2 gives

$$Q = E[ww^T] = E\left[\int_0^T\int_0^T \Phi(\tau_1)w_c(\tau_1)w_c^T(\tau_2)\Phi^T(\tau_2)d\tau_1 d\tau_2\right]. \tag{6.56}$$

Moving the expectation inside the integral and removing non-random processes

$$Q = \int_0^T\int_0^T \Phi(\tau_1)E\left[w_c(\tau_1)w_c^T\right](\tau_2)\Phi^T(\tau_2)d\tau_1 d\tau_2\,, \tag{6.57}$$

and substituting Equation 6.54 into Equation 6.57 yields

$$Q = \int_0^T\int_0^T \Phi(\tau_1)Q_c(\tau_1)\delta(\tau_1-\tau_2)\Phi^T(\tau_2)d\tau_1 d\tau_2\,. \tag{6.58}$$

According to Maybeck[27], this reduces to

$$Q = \int_0^T \Phi(\tau)Q_c(\tau)\Phi^T(\tau)d\tau\,. \tag{6.59}$$

Given the Gaussian white process noise strength, Equation 6.59 can be used to convert it to the form needed in the discrete Kalman filter (Equation 6.39).

6.9 Nonlinear Measurements

Often, the exact Cartesian position coordinates cannot be measured directly, or even as a linear combination. The Kalman filter can be easily adapted to account for such nonlinearities. The measurement equation changes from the linear form, given in Equation 6.3, to the nonlinear form,

$$\mathbf{y}_k = m(\tilde{\mathbf{x}}_k) + v\,, \tag{6.60}$$

where $m(\tilde{\mathbf{x}}_k)$ is the vector containing nonlinear functions of the state error. The values of $m(\tilde{\mathbf{x}}_k)$ come directly from the sensor measurements. An estimated measurement is calculated by

$$\hat{\mathbf{y}}_k = m(\hat{\mathbf{x}}_k^-). \tag{6.61}$$

The estimate update equation (Equation 6.17) is modified to take into account the nonlinearities, by

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + L_k(\mathbf{y}_k - \hat{\mathbf{y}}_k). \tag{6.62}$$

Note that Equations 6.62 and 6.17 are identical for the linear case when

$$\hat{\mathbf{y}}_k = C\hat{\mathbf{x}}_k^-. \tag{6.63}$$

Usually the state vector is used in the measurement rather than the state error vector,

$$\mathbf{y}_k = m(\mathbf{x}_k) + v, \tag{6.64}$$

where

$$\mathbf{x}_k = \tilde{\mathbf{x}}_k + \mathbf{x}_k^{ref}. \tag{6.65}$$

The subtraction in Equation 6.62 ensures that the result is the same. However, special consideration must be given to the measurement estimate, Equation 6.61. The substitution,

$$\hat{\mathbf{y}}_k = m(\overline{\mathbf{x}}_k^-), \tag{6.66}$$

where

$$\overline{\mathbf{x}}_k^- = \hat{\mathbf{x}}_k^- + \mathbf{x}_k^{ref} \tag{6.67}$$

must be made.

The Kalman gain (Equation 6.32) is a function of the linear coefficient matrix, C. In the nonlinear case, C can be replaced with H, where

$$H = \left.\frac{dm}{d\tilde{\mathbf{x}}_k}\right|_{\hat{\mathbf{x}}_k^-} \tag{6.68}$$

or

$$H = \left.\frac{dm}{d\mathbf{x}_k}\right|_{\bar{\mathbf{x}}_k^-}. \tag{6.69}$$

H is the derivative of the nonlinear vector function, m, with respect to the state or state error vector, evaluated at the *a priori* estimate. When H is calculated, the *a priori* estimate is the best known estimate at the time. For the linear case,

$$H = \left.\frac{dm}{d\tilde{\mathbf{x}}_k}\right|_{\hat{\mathbf{x}}_k^-} = C. \tag{6.70}$$

Calculating H requires taking the derivative of a vector with respect to another vector. If there are two vectors

$$\mathbf{a} = \begin{bmatrix} a_1 & \cdots & a_m \end{bmatrix}^T \tag{6.71}$$

and

$$\mathbf{b} = \begin{bmatrix} b_1 & \cdots & b_n \end{bmatrix}^T, \tag{6.72}$$

then

$$\frac{d\mathbf{a}}{d\mathbf{b}} = \begin{bmatrix} \frac{\partial a_1}{\partial b_1} & \frac{\partial a_1}{\partial b_2} & \cdots & \frac{\partial a_1}{\partial b_n} \\ \frac{\partial a_2}{\partial b_1} & \frac{\partial a_2}{\partial b_2} & \cdots & \frac{\partial a_2}{\partial b_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial a_m}{\partial b_1} & \frac{\partial a_m}{\partial b_2} & \cdots & \frac{\partial a_m}{\partial b_n} \end{bmatrix}. \tag{6.73}$$

Modifying the Kalman gain, Equation 6.32, to include the first-order approximation of the nonlinearities, results with

$$L_k = P_k(-)H^T\left(R + HP_k(-)H^T\right)^{-1}. \tag{6.74}$$

Also, to compute the updated estimation error covariance the same substitution is made

$$P_k(+) = (I - L_k H)P_k(-). \tag{6.75}$$

6.10 Range, Azimuth, and Elevation

For satellites, a ranging system that gives the scalar distance between the satellite and Earth, or two satellites, is common. In addition to the range, two angles, azimuth and elevation, are also measured. With these three measurements, the position can be determined.

For the drone satellites, positions are determined relative to the hub satellite. Range is scalar, so whether it comes from a sensor on the hub or the drone is unimportant. Azimuth and elevation, on the other hand, usually come from sensors on the drone spacecraft. They are in the frame of reference of the local coordinate system centered on the drone. Because the states of the drone spacecraft are represented with respect to the hub spacecraft, and the angles are with respect to the drone spacecraft, a coordinate transformation is required. However, if the local coordinate system on the drone is oriented the same as the reference coordinate system on the hub (b_1 is aligned with x, b_2 is aligned with y, and b_3 is aligned with z), the position of the drone can be determined from simple trigonometry.

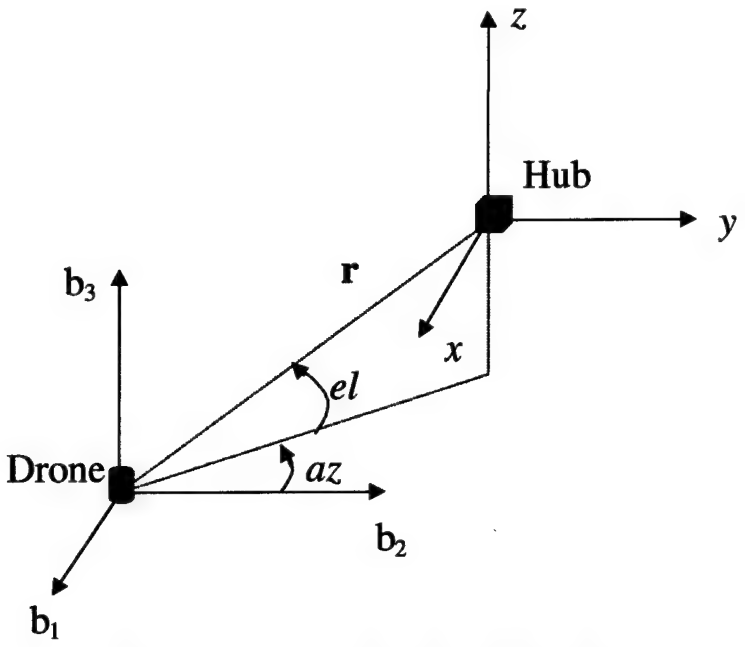

Figure 4. Range, azimuth, and elevation

For convention positive azimuth is defined as counter-clockwise from the b_2 direction in the b_1-b_2 plane. Positive elevation is defined from the b_1-b_2 plane upwards in the positive b_3 direction. The position of the drone relative to the hub is then

$$x = \mathbf{r}\cos(el)\sin(az) \tag{6.76}$$

$$y = -\mathbf{r}\cos(el)\cos(az) \tag{6.77}$$

$$z = -\mathbf{r}\sin(el), \tag{6.78}$$

where $\mathbf{r}$ is the range, *el* is the elevation, and *az* is the azimuth.

For one spacecraft,

$$m = \begin{bmatrix} \mathbf{r} & el & az \end{bmatrix}^T, \tag{6.79}$$

where

$$\mathbf{r} = \sqrt{\begin{bmatrix} x & y & z \end{bmatrix}^T \begin{bmatrix} x \\ y \\ z \end{bmatrix}} = \sqrt{x^2 + y^2 + z^2}, \tag{6.80}$$

$$el = \sin^{-1}\left(\frac{-z}{\mathbf{r}}\right), \tag{6.81}$$

and

$$az = \sin^{-1}\left(\frac{x}{\mathbf{r}\cos(el)}\right). \tag{6.82}$$

Clearly m is a nonlinear function of x, y, and z. From Equation 6.69 and Equation 6.73,

$$H = \left[\begin{array}{cccccc} \frac{\partial \mathbf{r}}{\partial x} & \frac{\partial \mathbf{r}}{\partial y} & \frac{\partial \mathbf{r}}{\partial z} & \frac{\partial \mathbf{r}}{\partial \dot{x}} & \frac{\partial \mathbf{r}}{\partial \dot{y}} & \frac{\partial \mathbf{r}}{\partial \dot{z}} \\ \frac{\partial el}{\partial x} & \frac{\partial el}{\partial y} & \frac{\partial el}{\partial z} & \frac{\partial el}{\partial \dot{x}} & \frac{\partial el}{\partial \dot{y}} & \frac{\partial el}{\partial \dot{z}} \\ \frac{\partial az}{\partial x} & \frac{\partial az}{\partial y} & \frac{\partial az}{\partial z} & \frac{\partial az}{\partial \dot{x}} & \frac{\partial az}{\partial \dot{y}} & \frac{\partial az}{\partial \dot{z}} \end{array}\right]_{\overline{\mathbf{x}}_k}. \tag{6.83}$$

Evaluating the partials,

$$\frac{\partial \mathbf{r}}{\partial x} = \frac{x}{\mathbf{r}} \tag{6.84}$$

$$\frac{\partial \mathbf{r}}{\partial y} = \frac{y}{\mathbf{r}} \tag{6.85}$$

$$\frac{\partial \mathbf{r}}{\partial z} = \frac{z}{\mathbf{r}} \tag{6.86}$$

$$\frac{\partial el}{\partial x} = \frac{xz}{\mathbf{r}^3\cos(el)} \tag{6.87}$$

$$\frac{\partial el}{\partial y} = \frac{yz}{\mathbf{r}^3\cos(el)} \tag{6.88}$$

$$\frac{\partial el}{\partial z} = -\frac{\left(\frac{1}{\mathbf{r}} - \frac{z^2}{\mathbf{r}^3}\right)}{\cos(el)} \tag{6.89}$$

$$\frac{\partial az}{\partial x} = \frac{\left(\frac{1}{\mathbf{r}\cos(el)} - \frac{x^2}{\mathbf{r}^3\cos(el)} - \frac{x^2 z^2}{\mathbf{r}^5\cos^3(el)} \right)}{\left(1 - \frac{x^2}{\mathbf{r}^2\cos^2(el)} \right)^{1/2}} \tag{6.90}$$

$$\frac{\partial az}{\partial y} = \frac{\left(\frac{-xy}{\mathbf{r}^3\cos(el)} - \frac{xyz^2}{\mathbf{r}^5\cos^3(el)} \right)}{\left(1 - \frac{x^2}{\mathbf{r}^2\cos^2(el)} \right)^{1/2}} \tag{6.91}$$

$$\frac{\partial az}{\partial z} = \frac{\left(\frac{-xz}{\mathbf{r}^3\cos(el)} - \frac{x\left(\frac{-2z}{\mathbf{r}^2} + \frac{2z^3}{\mathbf{r}^4} \right)}{2\mathbf{r}\cos^3(el)} \right)}{\left(1 - \frac{x^2}{\mathbf{r}^2\cos^2(el)} \right)^{1/2}} \tag{6.92}$$

and

$$\frac{\partial \mathbf{r}}{\partial \dot{x}} = \frac{\partial \mathbf{r}}{\partial \dot{y}} = \frac{\partial \mathbf{r}}{\partial \dot{z}} = \frac{\partial el}{\partial \dot{x}} = \frac{\partial el}{\partial \dot{y}} = \frac{\partial el}{\partial \dot{z}} = \frac{\partial az}{\partial \dot{x}} = \frac{\partial az}{\partial \dot{y}} = \frac{\partial az}{\partial \dot{z}} = 0. \tag{6.93}$$

For multiple spacecraft, H is appended in a block diagonal fashion, consisting of the H's from the individual satellites.

For the hub satellite, the same method works, assuming that its local coordinate frame from which azimuth and elevation are measured is oriented identically to the coordinate system the motion is based on. For the circular restricted three-body problem, the libration point would act as the "hub," and the hub spacecraft would act as a "drone." Realistically, sensors cannot detect the libration point, so the origin of the local coordinate system is translated in the negative radial direction to the earth. Thus, the earth acts as the "hub" and the hub spacecraft would act as a "drone." Because the local

coordinate system undergoes only translation, the dynamics remain the same, but initial conditions must be altered appropriately.

For the local coordinate system centered on the drone spacecraft to always be oriented the same as the hub coordinate system, spacecraft attitude is not considered. Realistically, the sensors measuring azimuth and elevation would be fixed at a certain location on the spacecraft. The attitude control system and the position control system would be tightly coupled.

7. STELLAR IMAGER

7.1 Mission Background

Stellar Imager (SI) is the concept for a space-based, UV-optical interferometer, proposed by Carpenter and Schrijver[28, 29]. The purpose of the mission is to view many stars with a sparse aperture telescope in an attempt to better understand how stars work. Hopefully, SI will further the understanding of the various effects of stars' magnetic fields, the dynamos that generate them, and the internal structures and dynamics of stars. With such information, a model could be developed to help forecast solar activity and understand the effects of stellar magnetic activity on astrobiology and life in the Universe. Such a model could also help us anticipate long-term Maunder minima and grand maxima occurrences of the sun, which can change the overall global temperatures causing crop failures. It could also help predict short-term solar activity (flares) that can disable satellites, knock out power grids, increase the speed of corrosion of oil pipelines, and jeopardize astronauts from particle radiation.

According to Rarogiewicz[30], an interferometer is an instrument that measures light waves by way of interference phenomena. Light from a single source is "picked off" by multiple flat (or spherical) mirrors, and then combined in such a way that they interfere with each other. From this induced interference, extremely small measurements are made. For SI, the light waves being utilized are in the ultra-violet frequency range.

SI has two primary science goals—imaging of stellar surface activity and asteroseismology. Currently, the sun is only one piece of data, and thus provides for insufficient constraints on theories of dynamos, turbulence, structure, and internal

mixing. More future missions are planned to find and image planets in other solar systems using IR-interferometry. SI will study the central stars of these solar systems to determine the impact on the habitability of the surrounding planets.

7.2 Performance Goals and Design Requirements

Carpenter[29] has outlined three primary performance goals: "1) Image a substantial sample of nearby dwarf and giant stars representing a broad range in magnetic activity, obtaining a resolution of order 1000 total pixels, 2) study a sample in detail, revisiting over many years, and measure sizes, lifetimes, and emergence patterns of stellar active regions, surface differential rotation, field dispersal by convective motions and meridional circulation, directly image the entire convection spectrum on giant stars and supergranulation, and 3) enable asteroseismology, using low to intermediate degree non-radial modes to measure internal stellar structure and rotation." The first goal states that the interferometer must view multiple stars, which implies that it can aim in different directions. The second goal forces the mission to have a lifetime of at least 10 years.

For stellar surface activity, ultra-violet images are necessary to differentiate between bright spots (strong magnetic fields) and the surrounding stellar surface. Integration time is the amount of time the interferometer must stay focused on the star to avoid smearing due to rotation, proper motions, and activity evolution. For surface imaging, the integration times will vary from hours for dwarfs, to days for giants. For asteroseismology, the integration times will vary from minutes for dwarfs, to hours for giants.

7.3 Preliminary Mission Design

A preliminary mission design for the SI mission was developed at the Integrated Mission Design Center (IMDC) at Goddard Space Flight Center. The leading concept for SI is a 500 meter diameter, Fizeau interferometer composed of 30 small drone satellites and one hub satellite. Each of these drone satellites will have a mirror that reflects the incoming light back to the hub. The Fizeau type refers to the method of combining the incoming light beams. After recombining the light beams into useful data, the hub will transmit this information back to Earth.

7.3.1 Formation Design

Through simulation, Allen and Rajagopal[29] determined that 30 drones with a maximum distance of 500 meters between any two elements would provide useful data. The following figure summarizes their efforts.

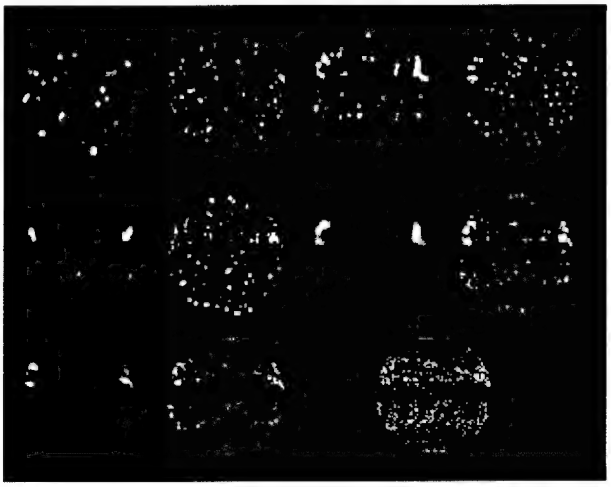

Figure 5. Simulated interferometric images of a sun-like star at 4 parsecs

The object on the bottom right is a model stellar image of a sun-like star 4 parsecs away. The other images are simulated results of different size and shape interferometers viewing the model. The first row has 6 elements, or drone satellites, and the second row

has 12. The satellites are arranged in a Y-formation. The third row has 30 elements arranged in a Golomb[31] rectangle configuration. The first and third columns have a maximum baseline, or distance between any two elements, of 250 meters. The second and fourth columns have a maximum baseline of 500 meters. The first two columns assume the formation is stationary and takes a snapshot of the model. The third and fourth columns assume the formation rotates 24 times in 15 degree increments. Several conclusions can be drawn from the figure. First, with more drone satellites, the interferometer recreates a better image of the model. Second, the larger the maximum baseline, the better the model is recreated. And third, the more snapshots that are taken, based on rotations of the formation, the more accurate the image. It has been determined that 30 drone satellites with a maximum baseline of 500 meters provide enough resolution for useful scientific data. However, for some stars, a second snapshot may be necessary, requiring the formation to reconfigure once with a 90 degree rotation.

A 30 element Golomb[31] rectangle can be imagined as a 30-by-30 non-diagonal matrix made up of zeros and ones, with ones representing elements. Every row and column has one and only one element in it. Although this explanation gives the image of the elements, or drones, lying in a plane, they actually lie on the surface of a sphere.

The hub satellite lies halfway between the surface of the sphere containing the drones and the origin. Focal lengths of both 0.5 km and 4 km are being considered. This would make the radius of the sphere either 1 km or 8 km. The following figures clarify the two design options.

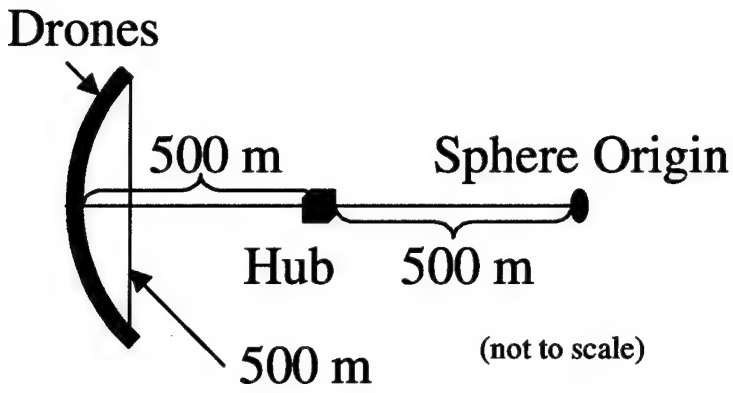

Figure 6. Satellite formation with focal length 0.5 km

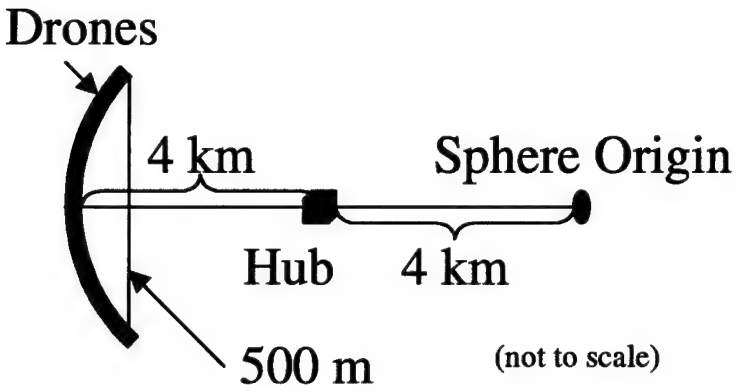

Figure 7. Satellite formation with focal length 4 km

7.3.2 Orbit Design

The type of orbit and location in space is an important part of mission design. For SI, the options considered were a low-Earth orbit, an Earth-following orbit, and a Lissajous orbit at a sun-Earth libration point. The strong gravity gradient for a low-Earth orbit causes increased difficulty for the formation flying precision necessary for SI. Also, scattered light from the proximity of the Earth would disrupt the interferometry. Because of the length of the mission (10 years minimum) and the large number of satellites (31), an individual satellite failure is probable. Replacements will be necessary and upgrades are likely. Re-stocking the formation in an Earth-following orbit is cost prohibitive. A similar problem exists for orbits around the L3, L4, and L5 sun-Earth libration points. At

L1, between the Earth and the sun, SI would have to deal with stray light coming from both the sun and the Earth. However, at L2, the sun and the Earth are on the same side of SI, reducing unwanted light and enhancing data collection. The best orbit choice for the formation is a Lissajous orbit around the sun-Earth L2 point. The amplitude of the Lissajous orbit will be about 600,000 km, but is not critical to the mission. With this orbit, SI will be able to cover the entire sky every half year while maintaining an aim perpendicular to the sun. For useful imaging, SI must aim within 10 degrees of perpendicular from the sun.

7.3.3 Control Design

To function properly, SI will need to accommodate a wide range of control functions. The formation must slew about the sky requiring movement of a few kilometers and attitude adjustments of up to 180 degrees. While imaging, though, the drones must maintain position within 3 nanometers of accuracy in the direction radial from the hub and within 0.5 millimeters of accuracy along the sphere surface. The accuracy required for attitude control while imaging is 5 milli-arcseconds tip and tilt (rotations out of the surface of the sphere). The rotation about the axis radial from the hub (rotation within the sphere) is a much less stringent 10 degrees.

Leitner and Schnurr[32] in the IMDC propose a three-tiered formation control approach. The first tier is the rough control using radio frequency (RF) ranging and modified star trackers for sensors and thrusters for actuators. The relative positions will be controlled to within a few centimeters. This level will drive lost-in-space emergencies, formation initialization, large translations due to formation slewing, collision avoidance, and Lissajous orbit maintenance. The modified star trackers will use

visual navigation techniques to initialize and coarsely maintain the formation. The RF ranging system will provide range measurements, and the modified star trackers will provide azimuth and elevation measurements. The actuators for this tier will be four low-thrust, high specific impulse (Isp) thrusters. The thrust level will be on the Newton to milli-Newton order of magnitude. This thesis is primarily concerned with the first-tier of the control structure.

The second tier is the intermediate control with a modulated laser ranging system as sensors and thrusters as actuators. The relative positions will be controlled to within 50 microns at this tier. This level will drive primary attitude adjustments and small translation maneuvers. Twelve 10-100 micro-Newton Indium Field Emission Electric Propulsion (FEEP) thrusters will be used for this level. This propulsion technology is currently available, but by 2015 should be vastly improved. Basically, this tier functions to smooth the transition from the rough control of the first-tier to the fine precision control of the third-tier.

The third-tier is the fine precision control. At this level, the satellites themselves will not move, but rather the mirrors will be adjusted by extremely accurate mechanical devices with an accuracy in the nanometer range. Rather than having a traditional sensor to determine measurements, phase diversity and wave-front error (WFE) sensing algorithms, using data from the incoming light beams, will determine the needed control. Currently, phase diversity and WFE sensing are in their infancy for use with spacecraft and formation flying concepts.

7.3.4 Other Subsystem Design Considerations

Current launch vehicles have the capability to deploy the hub and drones to the near L2. The preferred option from the IMDC is a Delta III 3940-11 to carry the hub and a Delta IV 4450-14 to carry all 30 drones and a dispenser. Both the dispenser and the hub would have a discardable hydrazine propulsion system to correct launch vehicle errors and aid in insertion into the Lissajous orbit. Initial total mass assumptions are 100 kg for each drone satellite, 550 kg for the hub, and 480 kg for the dispenser. The launch vehicles mentioned can easily accommodate such masses with plenty of margin.

The hub-drone setup lends itself to a centralized method of communications and data handling. The drones would talk to the hub, and the hub would process information and talk to the Earth. Relaying the information from the hub to the Earth would be enhanced by a central communications satellite at L2 for all missions flying there. The computer processing power required for a centralized approach would be immense, but may not be a problem by 2015. Estimated power requirements could be handled by body-mounted solar arrays on both the drones and the hub. The only challenge for the thermal system would be holding the mirror temperatures constant. With precision needed at the nanometer level, it is imperative to keep the mirrors isothermal to avoid contraction or expansion due to heating.

8. SIMULATION

8.1 Background

An essential design consideration for Stellar Imager is how much on-board fuel is required to complete the mission. This thesis focuses on the preliminary analysis of the position control of the formation to determine the fuel requirements. All spacecraft are considered "black boxes," and satellite attitudes are neglected. All satellites in the formation are considered identical, so Equations 5.9-5.12 are valid. Also, the drones are all controlled with respect to the hub, so the coupled satellite control equations in Section 5.2 are used. Three different scenarios make up the position control problem—maintaining the Lissajous orbit, slewing the formation to aim at another star, and reconfiguring the formation to take another snapshot of a star when necessary. These three scenarios are treated independently.

8.1.1 Earth-Sun L2 Circular Restricted Three-Body Dynamics

Specific numerical values are needed to build the dynamics matrix in Equation 2.44 for motion of a satellite in the vicinity of the sun-earth L2 point. Table 1 summarizes the constant parameters in the sun-earth system, given by the Astronomical Almanac[33].

Parameter	Equation	Value
Mass of the sun (m_{sun})	N/A	1.9891e30 kg
Mass of the earth (m_{earth})	$\frac{m_{sun}}{332946.0}$	5.9742e24 kg
Mass of the moon (m_{moon})	$m_{earth} * 0.01230002$	7.3483e22 kg
Universal Gravitation Constant (G)	N/A	0.4980621312 $\frac{m^3}{kg - day^2}$
Astronomical Unit (AU and D)	N/A	1.49597870e11 m
Distance from system barycenter to sun (D_1)	$\frac{m_{earth} + m_{moon}}{m_{sun} + m_{earth} + m_{moon}} * AU$	4.54841e5 m
Distance from system barycenter to the earth-moon barycenter (D_2)	$\frac{m_{sun}}{m_{sun} + m_{earth} + m_{moon}} * AU$	1.49597e11 m
Angular velocity of system (ω)	$\sqrt{\frac{G*(m_{sun} + m_{earth} + m_{moon})}{AU^3}}$ (Equation 2.6)	0.017202 $\frac{rad}{day}$

Table 1. Constant parameters of the sun-earth circular system

Based on the angular velocity of the system, the period of rotation comes out to be 365.26 days, or one year, as expected.

Szebehely[1] gives the location of the L2 point (which he calls L1) for the sun-earth system. Note that the 'x' direction of his rotating coordinate frame is opposite of mine, leading to the negative value of X_0. From this information, the partial derivatives in Equations 2.34-2.36 can be calculated. The results are summarized in Table 2.

Parameter	Value
X_0	$AU * -1.0100701938\,\text{m}$
Y_0	0
Z_0	0
U_{XX}	$0.00263065454908155\ \frac{rad}{day^2}$
U_{YY}	$-0.000871456562957885\ \frac{rad}{day^2}$
U_{ZZ}	$-0.00116737037068138\ \frac{rad}{day^2}$

Table 2. Constant parameters of the sun-earth L2 point

Using the above values, the dynamics matrix can now be determined

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ U_{XX} & 0 & 0 & 0 & 2\omega & 0 \\ 0 & U_{YY} & 0 & -2\omega & 0 & 0 \\ 0 & 0 & U_{ZZ} & 0 & 0 & 0 \end{bmatrix}. \tag{2.44}$$

Additional parameters are needed to determine the quasi-periodic reference orbit described by Equations 2.62-2.64. Table 3 lists these parameters.

Parameter	Value
In-plane frequency (ω_{xy})	0.0354038676524272 $\frac{rad}{day}$
Out-of-plane frequency (ω_z)	0.0341668021723043 $\frac{rad}{day}$
Non-oscillatory poles nulling factor (κ)	3.18878901709247

Table 3. Parameters for quasi-periodic orbit about sun-earth L2

The in-plane period is 177.47 days, and the out-of-plane period is 183.90 days. Every year, the formation makes roughly two revolutions both in-plane and out-of-plane.

The reference initial conditions listed in Table 4 correspond to the quasi-periodic Lissajous reference trajectory for one year shown in Figure 8.

Initial condition	Value
$x_{ref}(0)$	0 km
$y_{ref}(0)$	600000 km
$z_{ref}(0)$	0 km
$\dot{x}_{ref}(0)$	6661.563 $\frac{km}{day}$
$\dot{y}_{ref}(0)$	0 $\frac{km}{day}$
$\dot{z}_{ref}(0)$	-20500.081 $\frac{km}{day}$

Table 4. Reference initial conditions corresponding to Figure 8

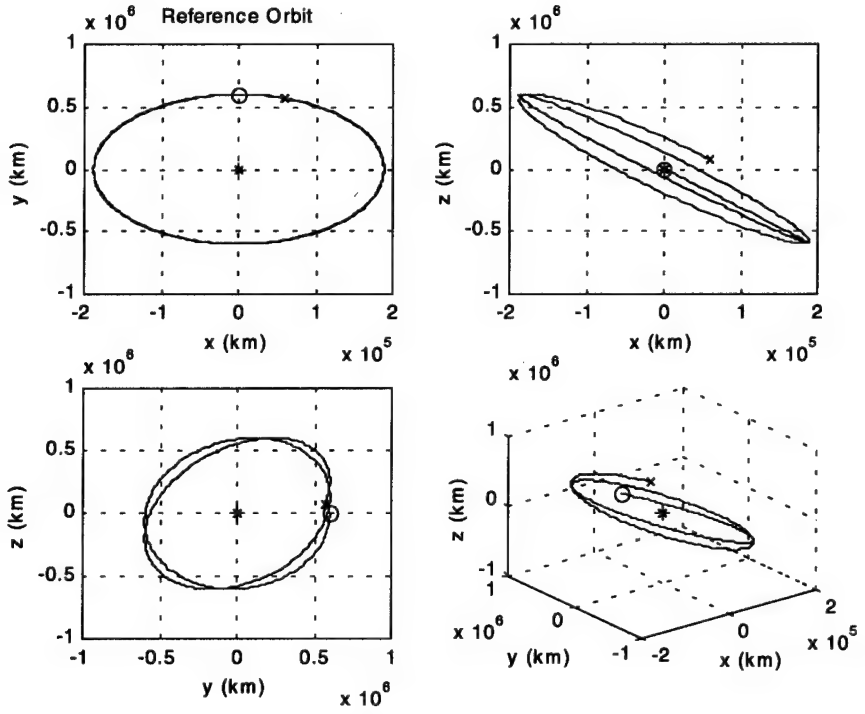

Figure 8. One year quasi-periodic reference orbit around L2

In Figure 8, the origin is the L2 point, the red circle is the starting reference position, and the red 'x' is the ending reference position after 365 days. The scale is in kilometers.

8.1.2 Generator

So far the dynamics of a satellite in the vicinity of the sun-earth L2 point have been approximated with the circular restricted three-body assumptions. These assumptions only account for gravitational forces from the sun and Earth. The moon is also included, but not as an independent body. The masses of the earth and moon are combined and assumed to be at the earth-moon barycenter. The motion of the sun and the earth-moon barycenter is also assumed to be circular around the system barycenter.

Realistically, the earth has a slightly elliptic orbit around the sun. The effect of the moon's gravity will vary depending on the exact position of the moon. Gravitational forces from other planets in the solar system affect the motion of satellites as well. The most significant non-gravitational force at the sun-earth L2 point is solar radiation

pressure, ignored in the circular restricted three-body problem. Dr. Kathleen Howell[34], from Purdue University, has a program named Generator that creates much more realistic Lissajous orbits than those derived from the circular restricted three-body problem. Using ephemeris files, Generator takes into account the effects of eccentricity, an independent moon, the other planets of the solar system, and solar radiation pressure. The resulting Lissajous orbit can then be used as a more accurate reference orbit. Another benefit of using a Generator derived reference orbit is that the origin of the rotating reference frame is based on the earth instead of the libration point. Thus, sensors can be modeled to measure range and angles from the hub satellite to the earth, rather than from the hub to the imaginary libration point.

With the initial conditions listed in Table 5, the Generator reference orbit is shown in Figure 9.

Initial condition	Value
$x_{ref}(0)$	1457251.466 km
$y_{ref}(0)$	572731.249 km
$z_{ref}(0)$	71916.835 km
$\dot{x}_{ref}(0)$	6779.986 $\frac{km}{day}$
$\dot{y}_{ref}(0)$	-5701.628 $\frac{km}{day}$
$\dot{z}_{ref}(0)$	79.552 $\frac{km}{day}$

Table 5. Reference initial conditions corresponding to Figure 9

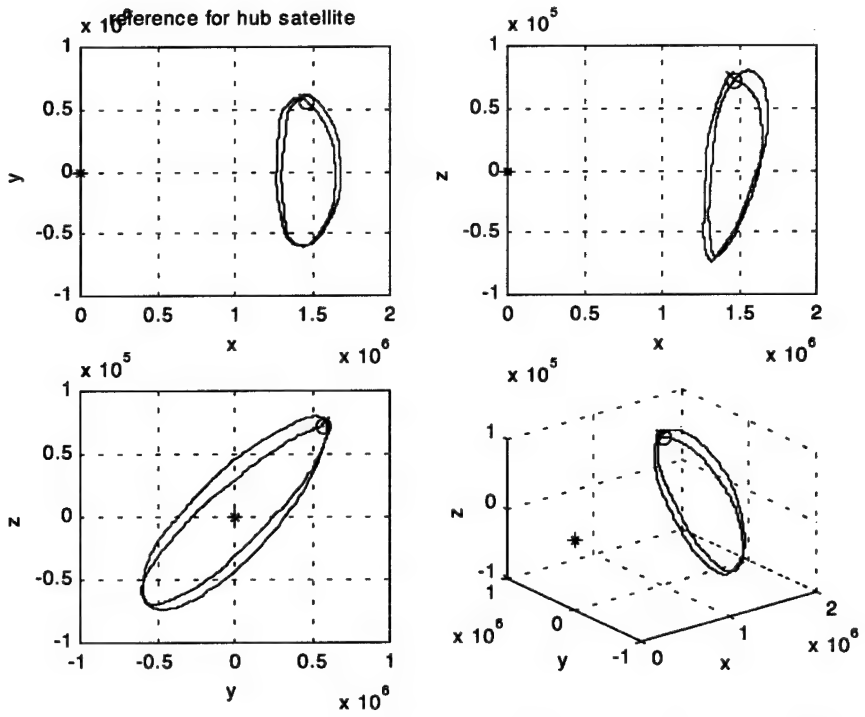

Figure 9. Generator based 359 day reference orbit around L2

The origin in Figure 9 is the earth. The reference orbit begins at the red circle and, after 359 days, ends at the red 'x'. The scale is in kilometers. The simulation length of 359 days is used to synchronize the simulation time with the output of Generator.

While using a Generator based reference orbit, the dynamics of Equation 2.44 (based on the circular restricted three-body problem) do not hold. However, in addition to providing the reference positions and velocities, Generator also numerically computes and outputs the dynamics matrix, *A*, for a single satellite at each epoch. Anderson[35] explains the transformation of the dynamics matrix from the inertial reference frame used in Generator to the outputted Earth-centered rotating frame used in the simulation. Special consideration must be given to the units of length and time. Generator uses velocities with units of meters per second. These must be converted to units of kilometers per day for use in the simulation. Also, the time unit within the dynamics matrix must be converted from seconds to days.

8.2 Lissajous Orbit Simulation

Following the Lissajous orbit is not a problem of formation control, but rather a problem of maintaining an orbit. Therefore, only the hub satellite needs simulation to determine the amount of control and fuel needed to maintain a Lissajous orbit. The results can be extended similarly to other satellites in the formation. The simulation does permit altering the number of satellites, but for each satellite, the time it takes to run the simulation increases proportionally.

8.2.1 Lissajous Orbit Simulation Development

First, the reference orbit is loaded and converted to the proper units from the Generator output. The Matlab file used to change the Generator reference orbit output to a form used in Matlab is given in Appendix A. Next, the initial conditions are defined. For this research, the satellite is assumed to start with no position or velocity errors. This means that the satellite initial conditions are identical to the reference initial conditions given in Table 5. In the simulation, the time step or maneuver interval, T, can varied by integer values of days, but is set at one day here.

At the beginning of each epoch, the corresponding single satellite dynamics matrix is taken from the Generator output and converted to proper units. The Matlab file used to change the Generator dynamics output to a form used in Matlab is given in Appendix B. The continuous control-mapping matrix used is the same as Equation 3.5. The state transition matrix is approximated by Equation 3.3, truncating after the quadratic term. The continuous state weighting matrix is chosen to be

$$W = \begin{bmatrix} 1e6 & & & & & & \\ & 1e6 & & & & & \\ & & 1e6 & & & & \\ & & & 1e3 & & & \\ & & & & 1e3 & & \\ & & & & & 1e3 \end{bmatrix}, \tag{8.1}$$

and the continuous control weighting matrix is chosen to be

$$V = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}. \tag{8.2}$$

The strength of the process noise is chosen as

$$Q_c = \begin{bmatrix} 0 & & & & & & \\ & 0 & & & & & \\ & & 0 & & & & \\ & & & 1e-6 & & & \\ & & & & 1e-6 & & \\ & & & & & 1e-6 \end{bmatrix}. \tag{8.3}$$

The strength of the process noise is set at a value large enough to be noticed, but not so much as to constrict or destabilize the system. The system is then discretized using Equations 4.14, 4.25, 4.26, 4.27, and 4.28. The discrete process noise covariance is calculated by Equation 6.59. The measurement noise covariance matrix for the hub satellite is chosen to be

$$R = \begin{bmatrix} 0.1^2 & & \\ & \left(\frac{0.3}{1500000}\right)^2 & \\ & & \left(\frac{0.3}{1500000}\right)^2 \end{bmatrix}. \tag{8.4}$$

The first term in the measurement noise covariance matrix assumes range measurements from the earth to the hub within 0.1 km or 100 meters. The second and third terms assume that the arc lengths corresponding to the azimuth and elevation angles are three times less accurate than the range measurements. To find the angular accuracy, simply divide the accuracy of arc length by the range. The L2 point is approximately 1.5 million kilometers from the earth and is used as a rough constant range.

If multiple satellites are simulated, then Equations 5.7, 5.20, and 6.7 are implemented. Also for multiple satellites, the discrete state weighting matrix, discrete control weighting matrix, discrete state-control couple weighting matrix, and discrete process noise covariance matrix are appended in block diagonal fashion--one block for each satellite, including the hub. The measurement noise covariance matrix is also appended block diagonally, but the covariance for the drone satellites is different from that of the hub. For drone satellites,

$$R = \begin{bmatrix} 0.0001^2 & & \\ & \left(\frac{0.0003}{0.5}\right)^2 & \\ & & \left(\frac{0.0003}{0.5}\right)^2 \end{bmatrix}. \tag{8.5}$$

Here, the range measurement is assumed to be accurate to within 0.1 meters, with three times less accurate arc lengths. The range from the hub to the drones is the focal length of the interferometer (either 0.5 or 4 km).

The control gain is calculated using the 'dlqr' command in Matlab, which solves Equations 4.55 and 4.56. The control is calculated by Equation 4.48, and the state error

and error estimate are propagated by Equations 6.1 and 6.8 respectively. The initial condition on the error estimate is set equal to the initial condition on the state error.

Equations 6.64-6.67 are used to process the measurements and measurement estimates. The measurements are assumed to be range, azimuth, and elevation, so Equations 6.79-6.82 are valid. The "true" measurements would realistically come from sensors, but for the simulation purposes, they are the known range, azimuth, and elevations with appended noise. Next, Equation 6.83 (filled in by Equations 6.84-6.93) is calculated. Finally, the Kalman filter algorithm is implemented with Equations 6.39, 6.74, 6.62, and 6.75 respectively. The initial estimation error covariance for the hub satellite is

$$P_1(+) = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 86.4^2 & & & \\ & & & & 86.4^2 & & \\ & & & & & 86.4^2 & \end{bmatrix}. \tag{8.6}$$

This assumes about 1 km accuracy for position (10 times greater than measurement noise covariance) and 1 meter per second for velocity. The 86.4 term is the conversion to kilometers per day from meters per second. For drone satellites, the tolerance is much more accurate due to the proximity from the drones to the hub as compared to the hub to the earth. Assuming position accuracy of 1 m and velocity accuracy of 1 millimeter per second, the initial estimation error covariance for drone satellites is

$$P_1(+) = \begin{bmatrix} .001^2 & & & & & \\ & .001^2 & & & & \\ & & .001^2 & & & \\ & & & .0864^2 & & \\ & & & & .0864^2 & \\ & & & & & .0864^2 \end{bmatrix}. \quad (8.7)$$

Finally, the whole process is repeated at each epoch, starting with loading the appropriate continuous dynamics matrix from Generator. The Lissajous orbit Matlab simulation is given in Appendix C.

8.2.2 Lissajous Orbit Simulation Results

The simulation of a satellite maintaining a Generator based Lissajous orbit around L2 provides three main results. The first is how well the satellite stayed its course, the second is how well the Kalman filter performed, and the third is how much fuel was necessary.

Figure 10 shows the tracking errors plotted against time for one simulation.

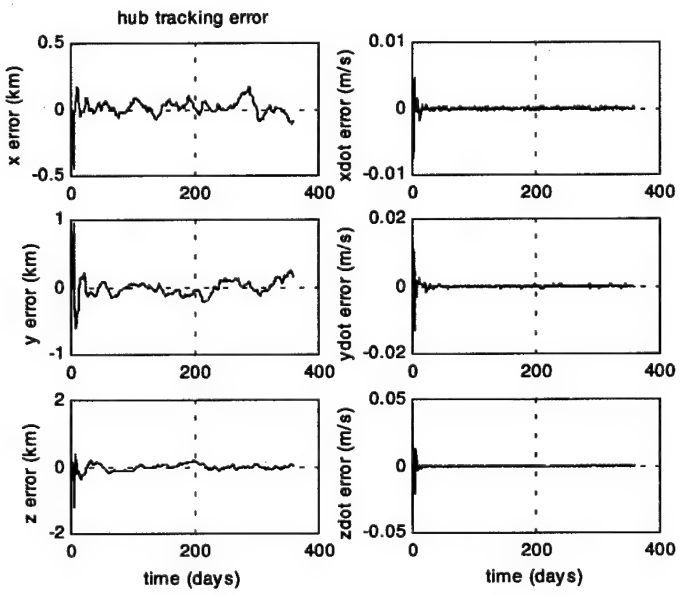

Figure 10. Hub tracking errors

The plots on the left are position tracking errors, and the plots on the right are velocity tracking errors. Running many simulations, it can be seen that the steady-state position tracking errors are within 0.25 km for each direction, and the steady-state velocity tracking errors are within 7.5e-4 meters per second for each direction.

Figure 11 shows the estimation errors of the hub satellite for one simulation.

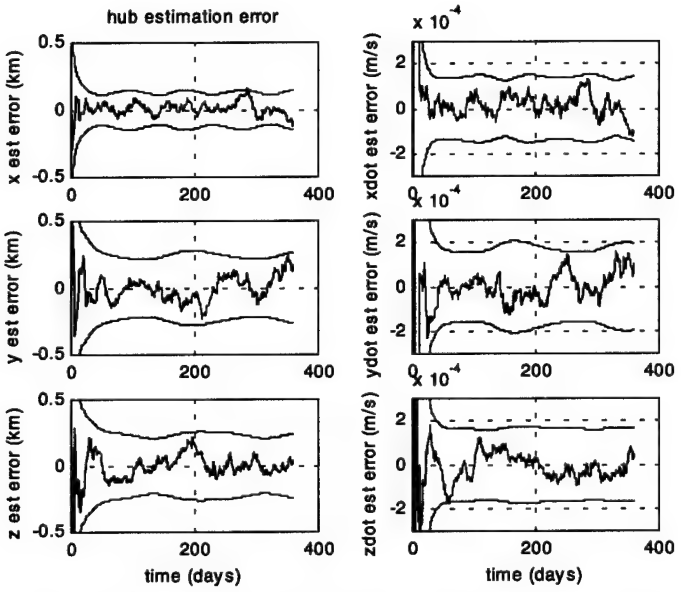

Figure 11. Hub estimation errors (one simulation)

The plots on the left are the position estimation errors, and the plots on the right are the velocity estimation errors. The blue lines represent the actual estimation errors, and the red lines represent the three-sigma value of the covariance. For any epoch, *k*, the three-sigma value of the covariance is calculated by

$$3\sigma_{kx} = \pm 3\sqrt{P_k(+)(1,1)} \tag{8.8}$$

$$3\sigma_{ky} = \pm 3\sqrt{P_k(+)(2,2)} \tag{8.9}$$

$$3\sigma_{kz} = \pm 3\sqrt{P_k(+)(3,3)} \tag{8.10}$$

$$3\sigma_{k\dot{x}} = \pm 3\sqrt{P_k(+)(4,4)} \tag{8.11}$$

$$3\sigma_{k\dot{y}} = \pm 3\sqrt{P_k(+)(5,5)} \tag{8.12}$$

$$3\sigma_{k\dot{z}} = \pm 3\sqrt{P_k(+)(6,6)}\,. \tag{8.13}$$

Because the estimation errors are based on randomness, many simulations must be run to determine useful results. Figure 12 shows the estimation errors of a dozen simulations.

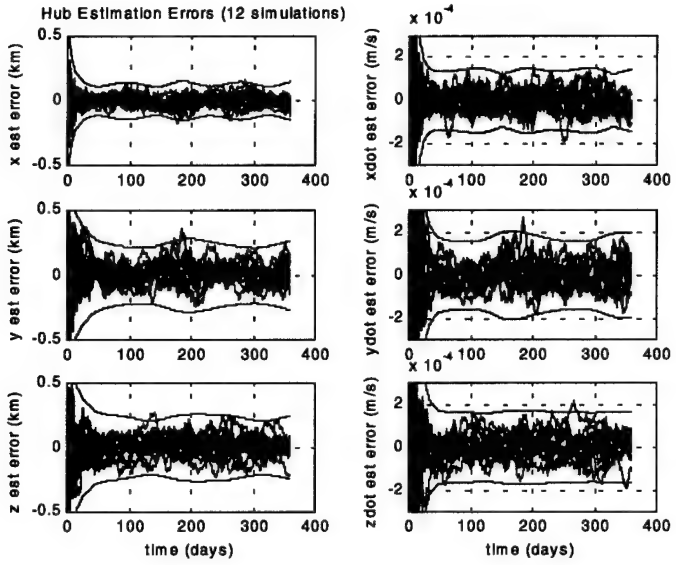

Figure 12. Hub estimation errors (12 simulations)

The estimation errors lie within the three-sigma values of the covariance with very few exceptions. The position estimation errors are within 250 meters, and the velocity estimation errors are within 2e-4 meters per second.

Figure 13 shows the control effort over the course of one simulation.

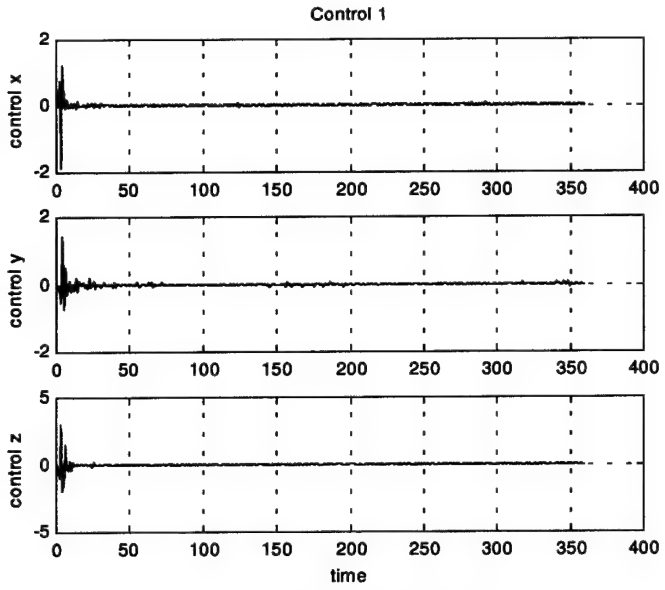

Figure 13. Hub control effort

Remember that the control is the applied directional accelerations. To determine the amount of fuel, the velocity change, or ΔV, is needed. The ΔV in each direction is found by numerical integration:

$$\Delta V_x = \sum_{k=1}^{359} |u_x| T \tag{8.14}$$

$$\Delta V_y = \sum_{k=1}^{359} |u_y| T \tag{8.15}$$

$$\Delta V_z = \sum_{k=1}^{359} |u_z| T, \tag{8.16}$$

where T is the maneuver interval. The absolute value of the control is taken because the direction of the maneuver has no bearing on the fuel used. The total ΔV for one simulation is calculated by

$$\Delta V = \Delta V_x + \Delta V_y + \Delta V_z. \tag{8.17}$$

Averaging the determined ΔV from a dozen simulations, the ΔV required to keep a satellite in a Lissajous orbit about L2 for 359 days is approximately 0.38 meters per second. Humble[36] gives the equation relating propellant mass to ΔV as

$$m_{prop} = m_0\left(1 - e^{\frac{-\Delta V}{Isp*g}}\right), \tag{8.18}$$

where m_{prop} is the propellant mass, m_0 is the initial spacecraft mass, Isp is the specific impulse of the thruster, and g is the gravitational acceleration constant 9.81 meters per second squared. From the Stellar Imager IMDC, Asato[37] gives the initial spacecraft mass to be 550 kg for the hub and 100 kg for a drone. Also, the Isp for the low-thrust, high-Isp thruster is assumed to be 10000 seconds. Therefore, the amount of propellant needed to maintain a Lissajous orbit for 359 days is less than 2.2 grams for the hub and less than 0.4 grams for each drone.

8.3 Formation Slewing

A key part of the SI mission is to image many stars. Following a Lissajous orbit around L2, SI could view the entire sky approximately every half-year while slewing about just the radial (x) axis. This will also maintain the aiming angle perpendicular to the sun. The formation slewing simulation follows a similar algorithm as the Lissajous orbit simulation.

8.3.1 Formation Slewing Simulation Development

First, the number of satellites to be simulated, the maneuver interval in minutes, and the simulation length in hours are specified. Also, variable are the place in the Lissajous orbit, ranging from the first day to the 358th, and the slew angle. For

continuity, the maneuver interval will be set to 1 minute and the length of the simulation to 24 hours. These 24 hours will occur from the second to the third day as modeled by Generator. Different slew angles will be examined.

The reference Lissajous orbit and dynamics are loaded from Generator, based upon the desired location in the orbit, and converted to the proper units. Similar to the Lissajous orbit simulation, the continuous control-mapping matrix used is Equation 3.5. The state transition matrix is approximated by Equation 3.3, truncating after the quadratic term. The continuous state weighting matrix is given in Equation 8.1, and the continuous control weighting matrix is given in Equation 8.2. The strength of the process noise is different from the Lissajous orbit simulation,

$$Q_c = \begin{bmatrix} 0 & & & & & \\ & 0 & & & & \\ & & 0 & & & \\ & & & 1e-24 & & \\ & & & & 1e-24 & \\ & & & & & 1e-24 \end{bmatrix}. \tag{8.19}$$

The strength of the process noise is set at a value large enough to be noticed, but not so much as to constrict or destabilize the system. The system is then discretized using Equations 4.14, 4.25, 4.26, 4.27, and 4.28. One key difference between the formation slewing simulation and the Lissajous orbit simulation is the maneuver interval or time step. In the Lissajous orbit simulation, the time step, T, is in days. In the formation slewing simulation, the time step, dT, is in minutes, and must be converted to days by dividing by 1440. This is then used for determining the state transition matrix and evaluating Equations 4.14, 4.25, 4.26, 4.27, and 4.28. The discrete process noise

covariance is calculated by Equation 6.59. The measurement noise covariance matrix is given by Equation 8.4 for the hub and Equation 8.5 for the drones.

The initial conditions of the Stellar Imager formation are coded into the simulation. The assumption is that the satellites start perfectly in formation, and slew to the new formation over the course of a day. The new slewed formation is the desired reference. The geometry of both the initial formation and slewed formation are explained in detail.

To build the initial formation, the hub spacecraft is assumed to be at the center of a sphere, on which the drones lie. The x direction of the hub-centered Cartesian coordinate system is parallel to and in the same direction as the radial axis of the earth-centered rotating coordinate system used by Generator. The y direction of the hub-centered system is parallel to and in the same direction as the along-track direction of the earth-centered system. The z direction of the hub-centered system is parallel to and in the same direction as the cross-track direction of the earth-centered system. Figure 14 shows the relationship between the two coordinate systems.

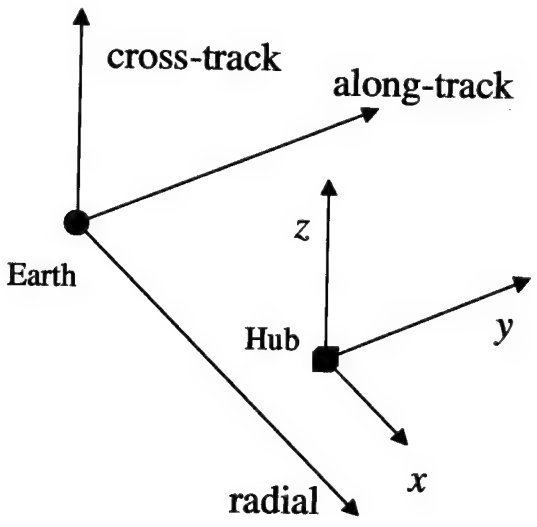

Figure 14. Hub-centered Cartesian coordinate system

From Figures 6 and 7, we see that the hub is not actually at the center of the sphere, but that will be taken into account later. The drones' positions are expressed in the spherical coordinates, r, θ, and ϕ, which relate to the hub-centered Cartesian coordinates by

$$x = r\sin(\theta), \tag{8.20}$$

$$y = r\cos(\theta)\sin(\phi), \tag{8.21}$$

and

$$z = r\cos(\theta)\cos(\phi). \tag{8.22}$$

The radial coordinate, r, is the distance from the hub to the drone and always equals the radius of the sphere, which is twice the desired focal length. The ϕ coordinate is measured from the positive z axis toward the positive y axis. Standing on the positive x axis and looking back, a positive rotation is clockwise. The θ coordinate is measured from the position in the y-z plane along the sphere in a clockwise direction when considered from the positive z axis looking back.

Initially, the center of the formation will be placed directly behind the hub in the along-track direction. Thus,

$$\theta^0_{cent} = 0 \tag{8.23}$$

and

$$\phi^0_{cent} = \frac{3\pi}{2}, \tag{8.24}$$

where θ^0_{cent} and ϕ^0_{cent} are initial coordinates corresponding to an imaginary central drone satellite. There are 30 drones for the SI mission, so 15 have a ϕ^0 coordinate greater than ϕ^0_{cent}, and 15 have a ϕ^0 coordinate less than ϕ^0_{cent}. Knowing that the maximum distance

between drones is 0.5 km, or 0.25 km to the imaginary central drone, the coordinates can be calculated by

$$\phi_i^0 = \phi_{cent}^0 + \frac{17-i}{15}\sin^{-1}\left(\frac{0.25}{r}\right), \tag{8.25}$$

where i represents the 2nd through 16th satellite (the hub being number 1), and

$$\phi_i^0 = \phi_{cent}^0 - \frac{i-16}{15}\sin^{-1}\left(\frac{0.25}{r}\right), \tag{8.26}$$

where i represents the 17th through 31st satellite. The θ^0 coordinates can be determined in a similar fashion, for i from 2 to 31,

$$\theta_i^0 = \theta_{cent}^0 + \frac{j}{15}\sin^{-1}\left(\frac{0.25}{r}\right), \tag{8.27}$$

where j is an integer between –15 and 15, such that each satellite, i, has a unique j. This formation is a Golomb[31] rectangle laid out on a spherical surface rather than a plane. The spherical coordinates for each drone and the imaginary central drone are intermediately transformed to Cartesian coordinates by Equations 8.20-8.22, yielding ${}^{int}x_i^0$, ${}^{int}y_i^0$, ${}^{int}z_i^0$, x_{cent}^0, y_{cent}^0, and z_{cent}^0. Finally, the satellites are translated by half of the imaginary central coordinates to account for the hub being halfway between the origin of the sphere and the surface of the sphere:

$$x_i^0 = {}^{int}x_i^0 - \frac{x_{cent}^0}{2} \tag{8.28}$$

$$y_i^0 = {}^{int}y_i^0 - \frac{y_{cent}^0}{2} \tag{8.29}$$

$$z_i^0 = {}^{int}z_i^0 - \frac{z_{cent}^0}{2}. \tag{8.30}$$

The drones initially start with zero relative velocity to the hub, so for i from 2 to 31,

$$\dot{x}_i^0 = \dot{y}_i^0 = \dot{z}_i^0 = 0. \tag{8.31}$$

The reference states are built by rotating the formation through a slew angle, α. The rotation is about the hub-centered x axis. Thus, for i from 2 to 31,

$$\phi_i^{ref} = \phi_i^0 - \alpha, \tag{8.32}$$

$$\theta_i^{ref} = \theta_i^0, \tag{8.33}$$

$$\phi_{cent}^{ref} = \phi_{cent}^0 - \alpha, \tag{8.34}$$

and

$$\theta_{cent}^{ref} = \theta_{cent}^0 = 0. \tag{8.35}$$

These spherical reference coordinates are plugged into Equations 8.20-8.22, which yields the intermediate ${}^{\text{int}}x_i^{ref}$, ${}^{\text{int}}y_i^{ref}$, ${}^{\text{int}}z_i^{ref}$, x_{cent}^{ref}, y_{cent}^{ref}, and z_{cent}^{ref}. Just as in Equations 8.28-8.30, the drones must be translated by half of the imaginary central drone position:

$$x_i^{ref} = {}^{\text{int}}x_i^{ref} - \frac{x_{cent}^{ref}}{2} \tag{8.36}$$

$$y_i^{ref} = {}^{\text{int}}y_i^{ref} - \frac{y_{cent}^{ref}}{2} \tag{8.37}$$

$$z_i^{ref} = {}^{\text{int}}z_i^{ref} - \frac{z_{cent}^{ref}}{2}. \tag{8.38}$$

No relative velocities between the hub and drones are desired, so for i from 2 to 31,

$$\dot{x}_i^{ref} = \dot{y}_i^{ref} = \dot{z}_i^{ref} = 0. \tag{8.39}$$

The drones' reference states do not change over the course of the simulation.

The hub satellite's initial and reference conditions are treated differently than the drones', because the hub's reference states change over the duration of the simulation. From Generator, the hub reference conditions are known at the beginning of the

simulation, $\mathbf{x}_1^{ref\,0}$, and at the end of simulation, one day later, $\mathbf{x}_1^{refend}$. The initial conditions are set equal to the known starting reference conditions,

$$\mathbf{x}_1^0 = \mathbf{x}_1^{ref\,0}. \tag{8.40}$$

The hub reference states at some epoch, k, during the simulation are determined by linear interpolation,

$$\mathbf{x}_1^{refk} = \left(\mathbf{x}_1^{refend} - \mathbf{x}_1^{ref\,0}\right)\frac{dT}{1440}k + \mathbf{x}_1^{ref\,0}. \tag{8.41}$$

The control and estimation algorithm from one epoch to another are identical to the Lissajous orbit simulation detailed at the end of Section 8.2.1. The lone difference is that the dynamics are assumed to be time-invariant over the course of the simulation rather than being updated at each epoch; therefore, the control gain is solved for once and held constant throughout the simulation. The formation slewing Matlab simulation is given in Appendix D.

8.3.2 Formation Slewing Simulation Results

The formation slewing simulation provides a plethora of useful results. The hub tracking error, estimation error, and control are found for focal lengths of both 0.5 and 4 km. Each drones' tracking error, estimation error, and control are determined as well. For conciseness, only the results for two drones will be shown (satellite numbers 2 and 31). Two slewing angles are investigated, 90 degrees and 30 degrees. Figure 15 provides a clear image of the entire SI formation slewing 90 degrees, with a 0.5 km focal length.

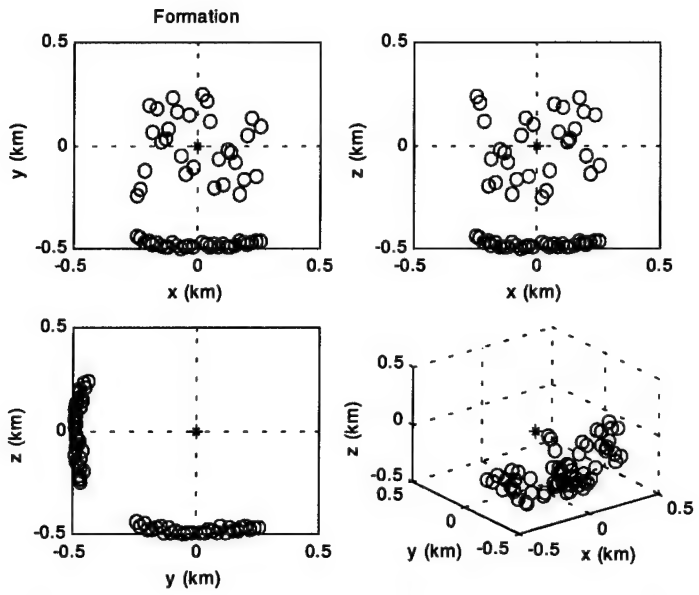

Figure 15. SI formation before and after 90 degree slew (0.5 km focal length)

The magenta circles represent drones at the beginning of the simulation, and the red circles represent drones at the end of the simulation. The hub is the black asterisk at the origin. The upper-right plot illustrates the Golomb[31] rectangle formation projected into the *x*-z plane. The lower-left plot clearly shows the drones slewing 90 degrees about the hub-centered *x* axis.

Figures 16-18 show the tracking errors for the hub and 2 drones. Although the plots are specific to a 90 degree slew with a 0.5 km focal length, they are qualitatively representative of all the different slew angle and focal length scenarios.

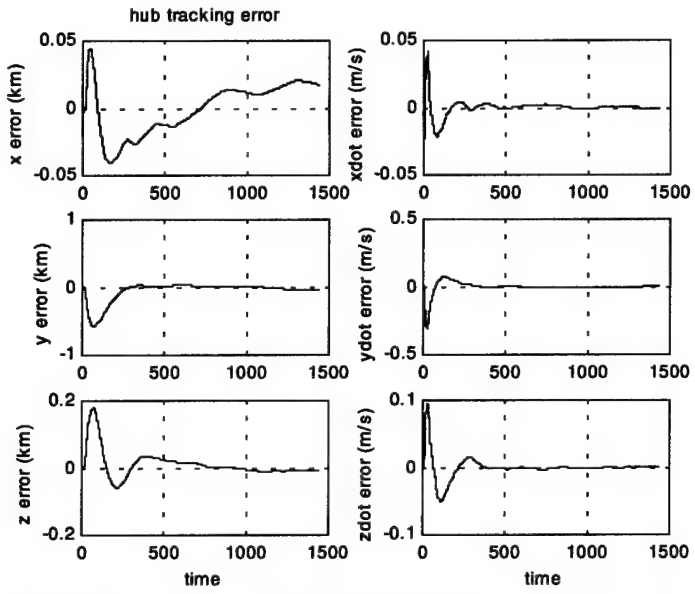

Figure 16. Hub tracking error (90 degree slew and 0.5 km focal length)

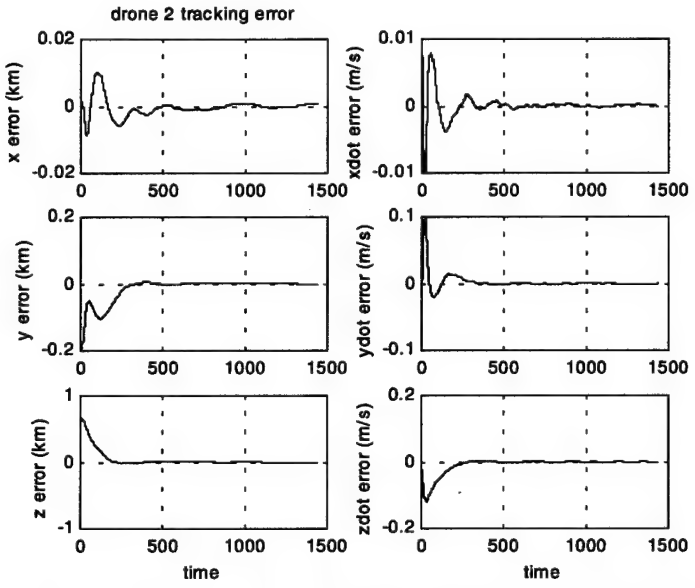

Figure 17. Drone 2 tracking error (90 degree slew and 0.5 km focal length)

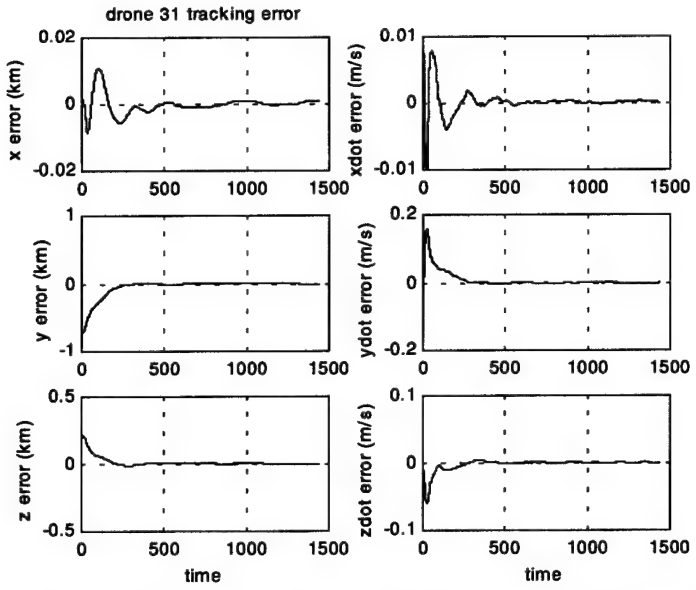

Figure 18. Drone 31 tracking error (90 degree slew and 0.5 km focal length)

For both 90 degree and 30 degree slews with either a 0.5 or 4 km focal length, the hub tracks to within 50 meters of its reference position and to within 5 millimeters per second of its desired velocity. The drones all track to within 3 meters of their desired reference positions and to within 1 millimeter per second of their desired zero velocities.

Table 6 lists the total position tracking error after one day for the different scenarios when noise is turned off.

Focal length (km)	Slew angle (deg)	Hub position tracking error (m)	Drone 2 position tracking error (m)	Drone 31 position tracking error (m)
0.5	90	8.33e-6	3.075e-6	3.075e-6
0.5	30	3.05e-6	1.126e-6	1.126e-6
4	90	7.35e-5	2.713e-5	2.713e-5
4	30	2.69e-5	9.93e-6	9.93e-6

Table 6. Formation slewing position tracking errors with no noise

For all scenarios, neglecting noise, the velocity tracking error is essentially zero after one day. If noise is turned off, the tracking errors go asymptotically to zero, as expected with

a linear quadratic regulator control strategy. Clearly, the noise and estimation errors have a significant effect on the tracking errors.

Figures 19-22 show the estimation error results of a dozen simulations for the hub with varying slew angles and focal lengths.

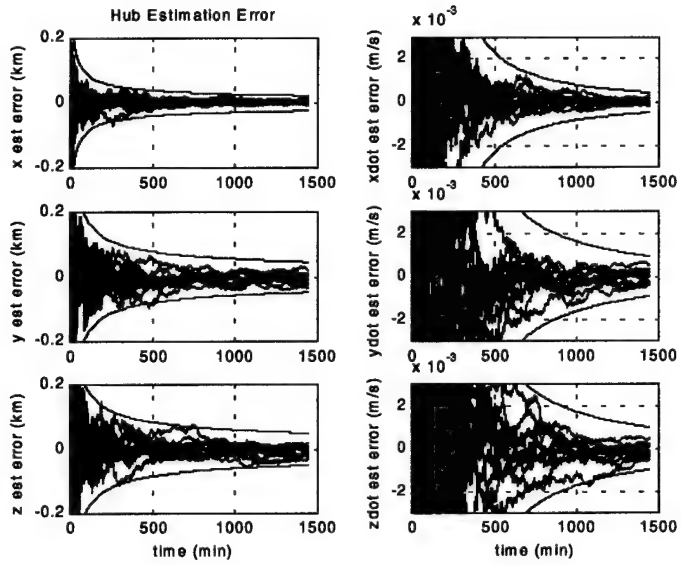

Figure 19. Hub estimation error for 0.5 km focal length and 30 degree slew

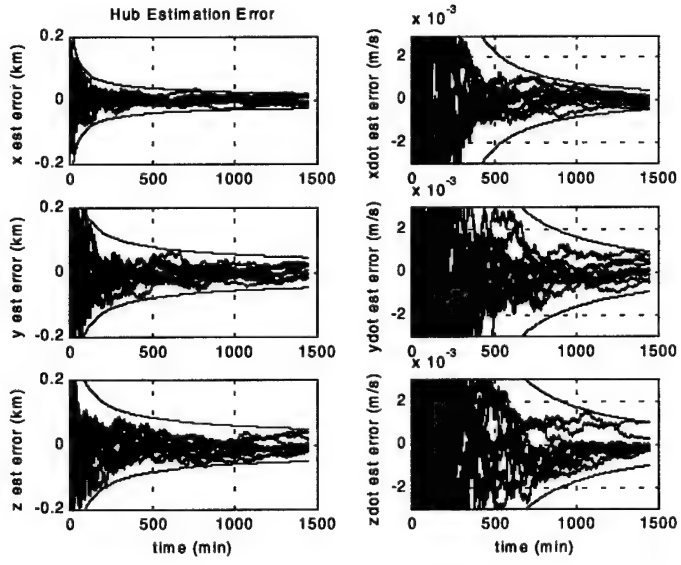

Figure 20. Hub estimation error for 0.5 km focal length and 90 degree slew

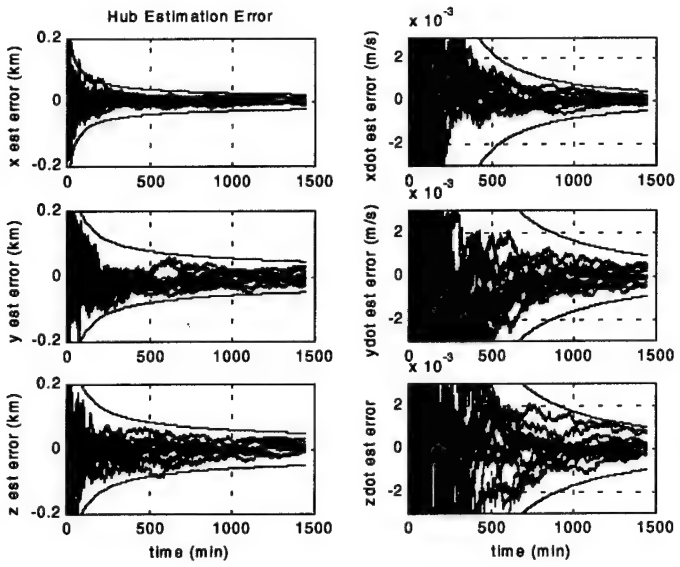

Figure 21. Hub estimation error for 4 km focal length and 30 degree slew

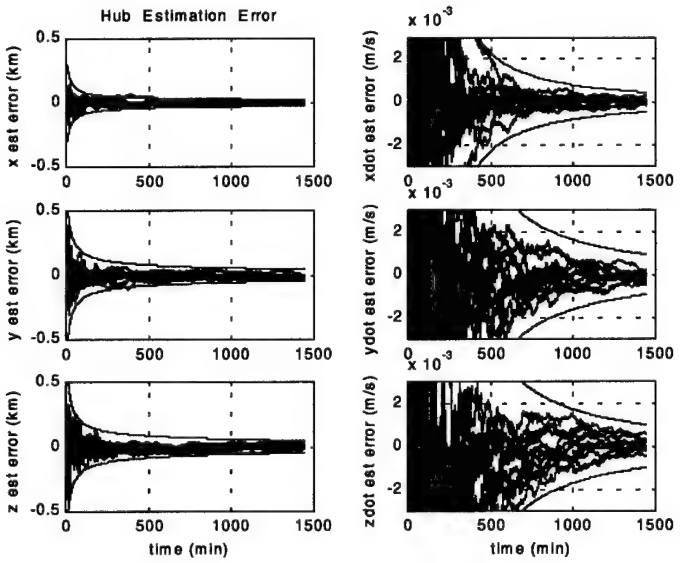

Figure 22. Hub estimation error for 4 km focal length and 90 degree slew

The red lines are the three-sigma values calculated by Equation 8.8-8.13. The hub's steady-state position three-sigma values are about 50 meters, and the steady-state velocity three-sigma values are about 1 millimeter per second for all scenarios. The estimation errors are within the three-sigma values with few exceptions. The three-sigma values change for each simulation (because the noise is random), but the change cannot be seen

for the hub because the order of magnitude of change is much, much less than their overall value.

Figures 23-26 show the estimation errors for a dozen simulations for the first drone satellite (satellite number 2).

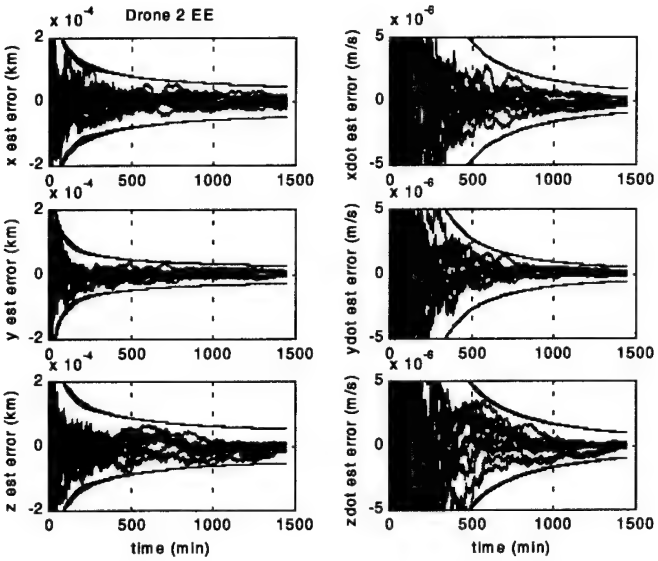

Figure 23. Drone 2 estimation errors for 0.5 km focal length and 30 degree slew

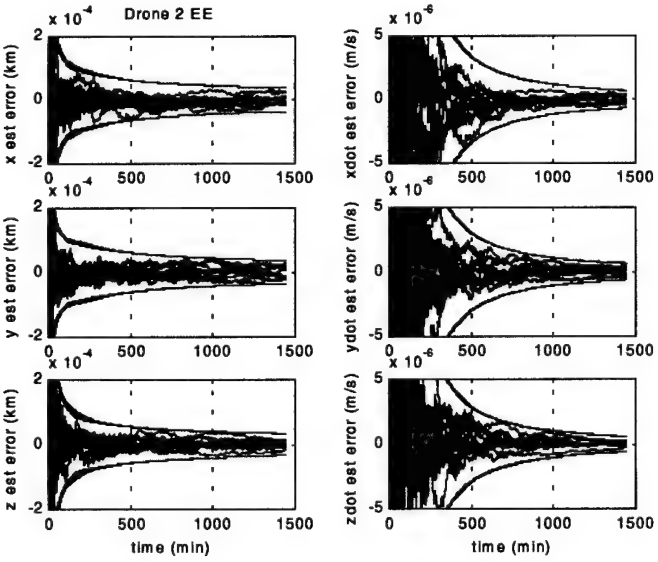

Figure 24. Drone 2 estimation errors for 0.5 km focal length and 90 degree slew

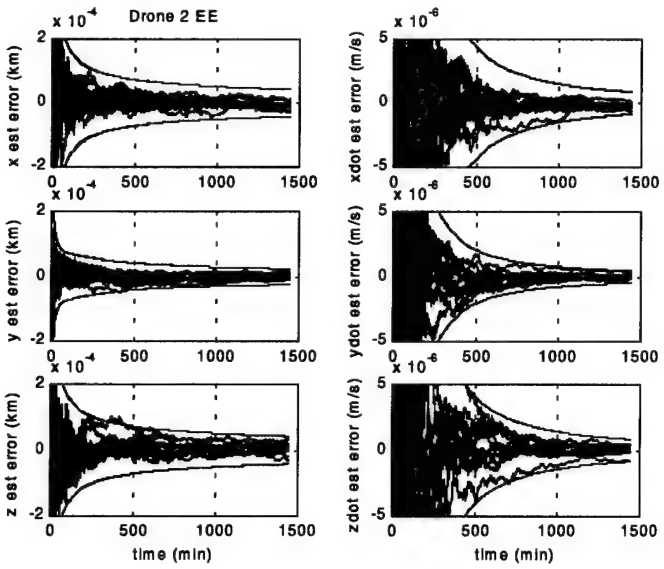

Figure 25. Drone 2 estimation errors for 4 km focal length and 30 degree slew

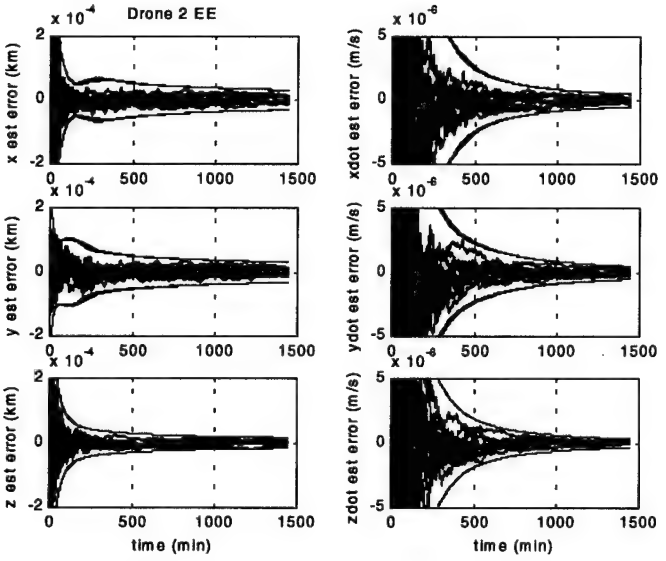

Figure 26. Drone 2 estimation errors for 4 km focal length and 90 degree slew

Drone 2's estimation errors are, with few exceptions, within the three-sigma values for all scenarios. The three-sigma value change from one simulation to another can be seen in the drone estimation error plots. The range from the hub to the drone is either 0.5 or 4 km, whereas the range from the hub to the earth is about 1.5 million km.

Figures 27-30 show the estimation errors of a dozen simulations for the last drone (satellite number 31).

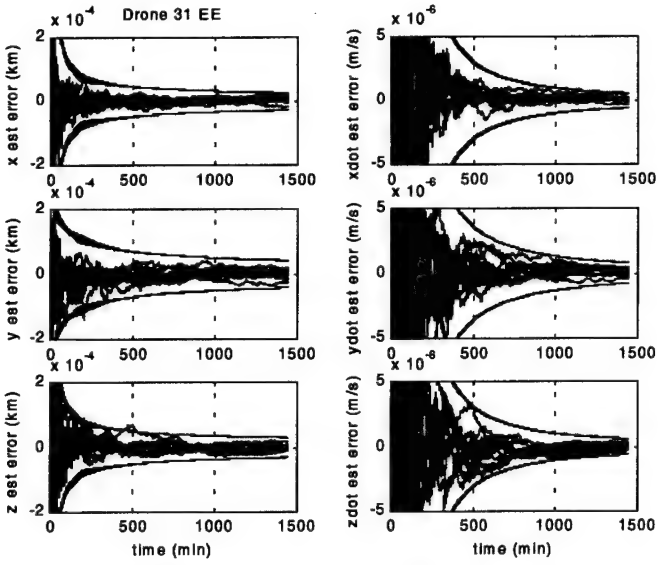

Figure 27. Drone 31 estimation errors for 0.5 km focal length and 30 degree slew

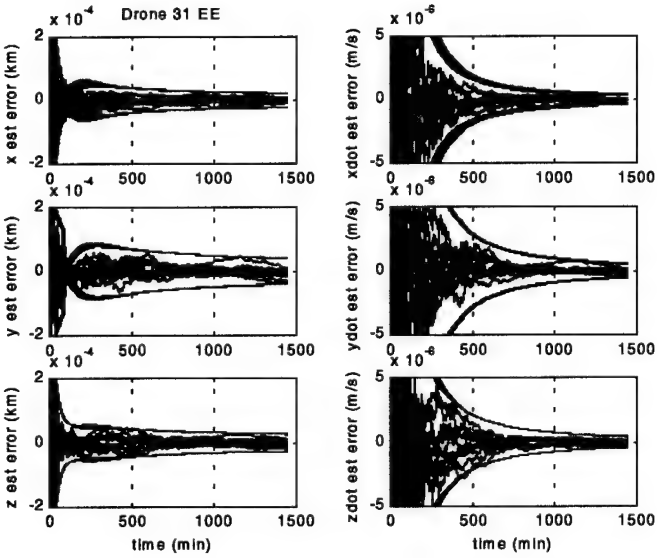

Figure 28. Drone 31 estimation errors for 0.5 km focal length and 90 degree slew

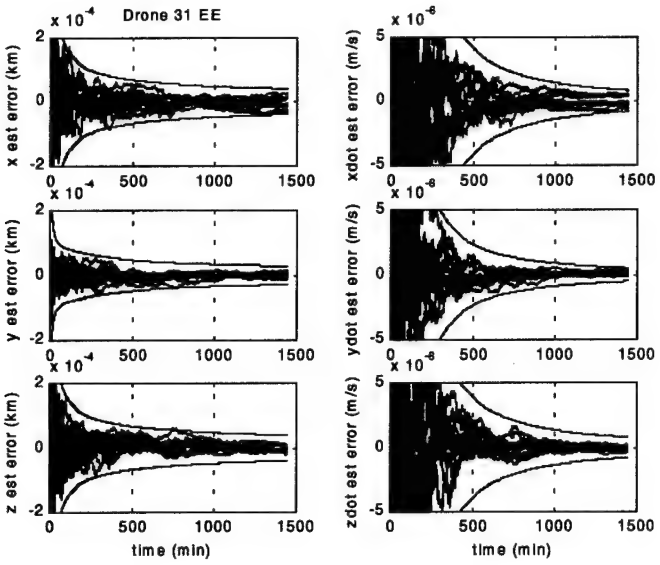

Figure 29. Drone 31 estimation errors for 4 km focal length and 30 degree slew

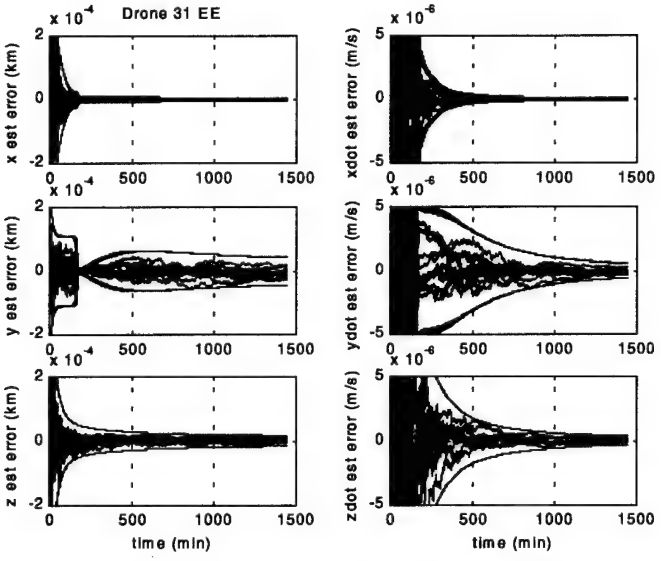

Figure 30. Drone 31 estimation errors for 4 km focal length and 90 degree slew

Drone 31's estimation errors are within the three-sigma values with few exceptions. The steady-state position three-sigma values are less than 0.1 meters for each drone in every scenario. The steady-state velocity three-sigma values are less than 1 micrometer per second for each drone in every scenario.

From the control efforts, the directional ΔV can be determined for each satellite:

$$\Delta V_x = \sum_{k=1}^{1440} |u_x| \frac{dT}{1440} \tag{8.42}$$

$$\Delta V_y = \sum_{k=1}^{1440} |u_y| \frac{dT}{1440} \tag{8.43}$$

$$\Delta V_z = \sum_{k=1}^{1440} |u_z| \frac{dT}{1440}. \tag{8.44}$$

The total ΔV is found from the directional ΔV's by Equation 8.17. The differences between Equations 8.14-8.16 and Equations 8.42-8.44 are the number of maneuvers over the course of the simulation and the maneuver interval. The formation slewing simulation runs for one day, with one maneuver per minute (1440 maneuvers), whereas the Lissajous orbit simulation runs for 359 days with one maneuver per day. Table 7 shows the average ΔV's for a dozen simulations for the various scenarios.

Focal Length (km)	Slew Angle (deg)	Hub ΔV (m/s)	Drone 2 ΔV (m/s)	Drone 31 ΔV (m/s)
0.5	30	1.0705	0.8271	0.8307
0.5	90	1.1355	0.9395	0.9587
4	30	1.2688	1.1189	1.1315
4	90	1.8570	2.1907	2.1932

Table 7. Average formation slewing ΔV's

The larger the angle the formation slews through, the more ΔV is needed. Also, the larger the focal length, the more ΔV required. Table 8 shows the corresponding propellant masses needed to achieve the ΔV's given in Table 7, with the assumptions that the Isp of the low-thrust thrusters is 10000 seconds, the initial mass of the hub is 550 kg, and the initial mass of each drone is 100 kg.

Focal Length (km)	Slew Angle (deg)	Hub m_{prop} (g)	Drone 2 m_{prop} (g)	Drone 31 m_{prop} (g)
0.5	30	6.0018	0.8431	0.8468
0.5	90	6.3662	0.9577	0.9773
4	30	7.1135	1.1406	1.1534
4	90	10.4112	2.2331	2.2357

Table 8. Average formation slewing propellant masses

When the noise is turned off, the required ΔV and propellant mass are reduced significantly. Table 9 shows the ΔV's and Table 10 shows the corresponding propellant masses when noise is removed.

Focal Length (km)	Slew Angle (deg)	Hub ΔV (m/s)	Drone 2 ΔV (m/s)	Drone 31 ΔV (m/s)
0.5	30	0.0504	0.0853	0.0998
0.5	90	0.1581	0.2150	0.2315
4	30	0.4420	0.5896	0.6441
4	90	1.3945	1.9446	1.9469

Table 9. Required ΔV's for formation slewing cases without noise

Focal Length (km)	Slew Angle (deg)	Hub m_{prop} (g)	Drone 2 m_{prop} (g)	Drone 31 m_{prop} (g)
0.5	30	0.2826	0.0870	0.1017
0.5	90	0.8864	0.2192	0.2360
4	30	2.4781	0.6010	0.6566
4	90	7.8182	1.9822	1.9846

Table 10. Required propellant masses for formation slewing cases without noise

8.4 Formation Reorientation

For some stars, one snapshot from the SI formation will not provide enough sampling data for sufficient resolution. In these cases, the drones must rotate 90 degrees and take another snapshot. In Section 8.3.1, an assumption is made that the drones are located behind the hub, in the negative y direction. Following this assumption, the 90 degree reorientation will be a rotation about the y axis. The aim from the drones, through the hub, to the desired star, is maintained with such a reorientation.

8.4.1 Formation Reorientation Simulation Development

The simulation development for the formation reorientation is nearly identical to the formation slewing simulation in Section 8.3.1. The simulation is set to run for one day with a maneuver interval of one minute. The place in the Lissajous orbit is variable. Generator gives the dynamics and hub reference for the corresponding place in the Lissajous orbit. The continuous state-weighting matrix, control-weighting matrix, and process noise strength are given in Section 8.3.1, as well as the method for discretizing and solving for the control gain. The control gain for the formation reorientation simulation is identical to the formation slewing simulation. The initial conditions are built through Equations 8.20-8.31.

The difference between the formation reorientation and formation slewing simulations is the desired reference states of the drones. The formation slewing reference states are based on a variable slew angle, α, rotation about the x axis, whereas the formation reorientation reference states are based on a 90 degree rotation about the y axis. The imaginary central drone has the same reference conditions as initial conditions,

$$\phi_{cent}^{ref} = \phi_{cent}^{0} = \frac{3\pi}{2} \tag{8.45}$$

and

$$\theta_{cent}^{ref} = \theta_{cent}^{0} = 0. \tag{8.46}$$

Setting

$$\phi_i^{ref} = -\theta_i^0 + \phi_{cent}^0 \tag{8.47}$$

and

$$\theta_i^{ref} = \phi_i^0 - \phi_{cent}^0, \tag{8.48}$$

for i from 2 to 31, gives the drones' reference positions. The rotation is 90 degrees counterclockwise about the y axis when looking back from the hub. Once again, zero relative velocity between the drones and the hub is desired, so Equation 8.39 holds.

The control and estimation algorithm from one epoch to another is identical to the formation slewing algorithm. The formation reorientation Matlab simulation is given in Appendix E.

8.4.2 Formation Reorientation Simulation Results

The formation reorientation simulation provides similar results as the Lissajous orbit and formation slewing simulations. The tracking errors, estimation errors, and control efforts of the hub and drones are determined. From the control effort, the required ΔV and corresponding propellant mass can be found. Figure 31 shows the first four drones of the formation before and after reorientation (with a 0.5 km focal length). Only four drones are pictured for clarity.

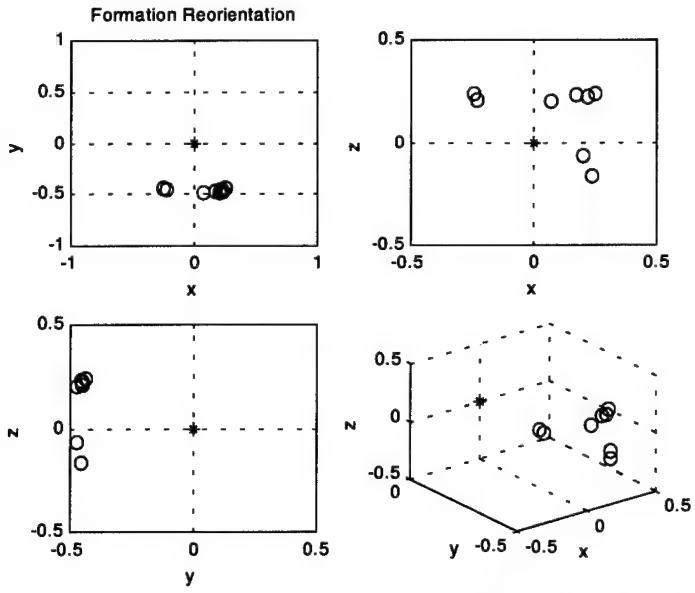

Figure 31. Formation reorientation (0.5 km focal length)

The scale is in kilometers in all four pictures. The plot in the upper-right shows the projection of the formation in the *x*-*z* plane. The magenta circles are the drones at their initial conditions, and the red circles are the drones after the simulation. The formation appears to have rotated clockwise 90 degrees about the *y* axis, but the hub is into the page, so the rotation is counterclockwise when viewed from the hub.

Figures 32-34 show the tracking errors for the hub, the first drone (#2), and the last drone (#31) for the 0.5 km focal length case.

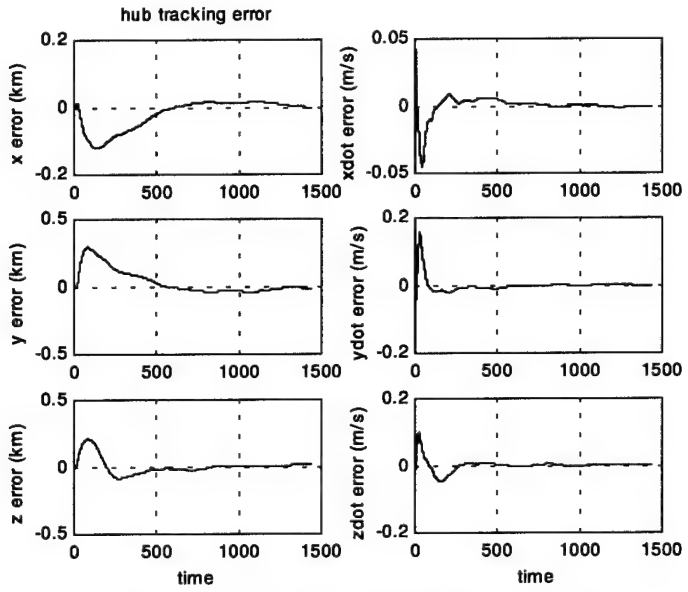

Figure 32. Hub tracking error (0.5 km focal length)

The hub tracks to within 40 meters of its reference position, and to within 8 millimeters per second of its reference velocity for both focal lengths.

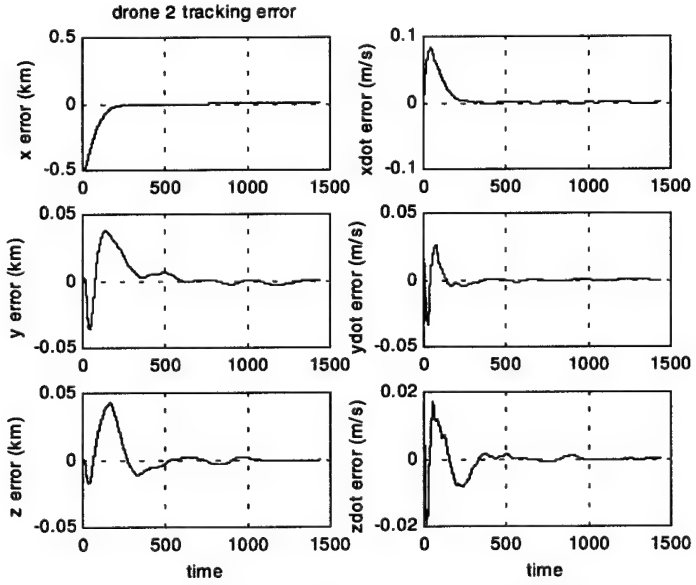

Figure 33. Drone 2 tracking error (0.5 km focal length)

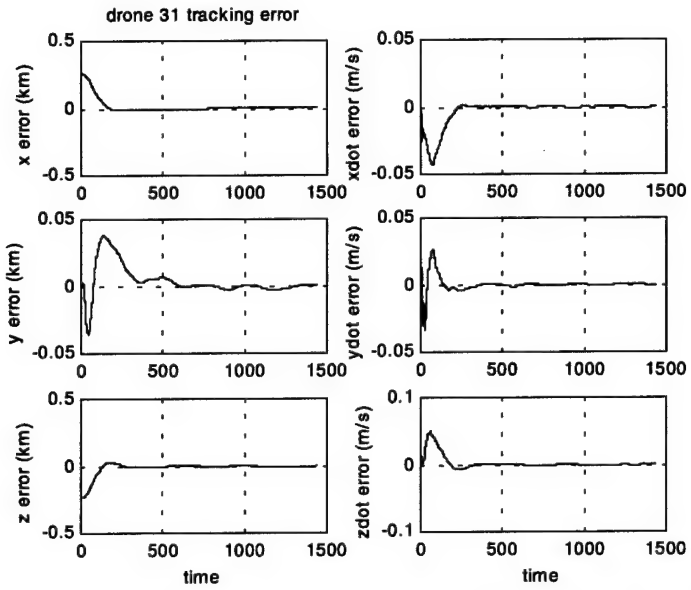

Figure 34. Drone 31 tracking error (0.5 km focal length)

Every drone tracks to within 4 meters of its reference position, and to within 1.5 millimeters per second of its desired zero velocity for both focal lengths.

When noise is turned off, the satellites track much better than when the noise is included. Table 11 shows the total position tracking errors for the various satellites after one day, when noise is eliminated.

Focal length (km)	Hub position tracking error (m)	Drone 2 position tracking error (m)	Drone 31 position tracking error (m)
0.5	2.147e-6	0.792e-6	0.793e-6
4	2.146e-6	0.792e-6	0.793e-6

Table 11. Formation reorientation position tracking errors with no noise

The tracking errors go asymptotically to zero, and the velocity tracking errors are essentially zero at the end of a day. Just as with the formation slewing simulation, the noise, and in turn the estimation errors, are the largest reason for imperfect tracking.

Figures 35 and 36 show the hub estimation errors for a dozen simulations with the different focal lengths.

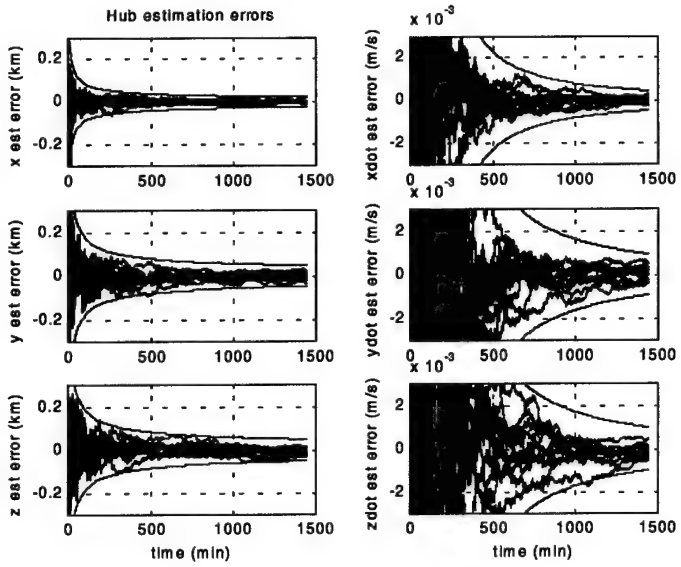

Figure 35. Hub estimation errors (0.5 km focal length)

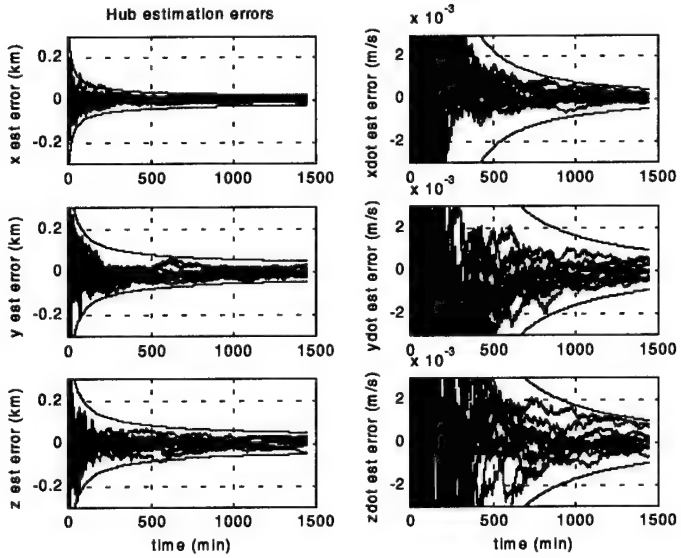

Figure 36. Hub estimation errors (4 km focal length)

The steady-state x three-sigma value is about than 30 meters. The steady-state y and z three-sigma value is about 50 meters. The steady-state velocity three-sigma values are about 1 millimeter per second. The hub estimation errors stay within the three-sigma values except for rare occasions.

Figures 37 and 38 show the estimation errors of a dozen simulations for the first drone (satellite #2), and Figures 39 and 40 show the estimation errors of a dozen simulations for the last drone (satellite #31).

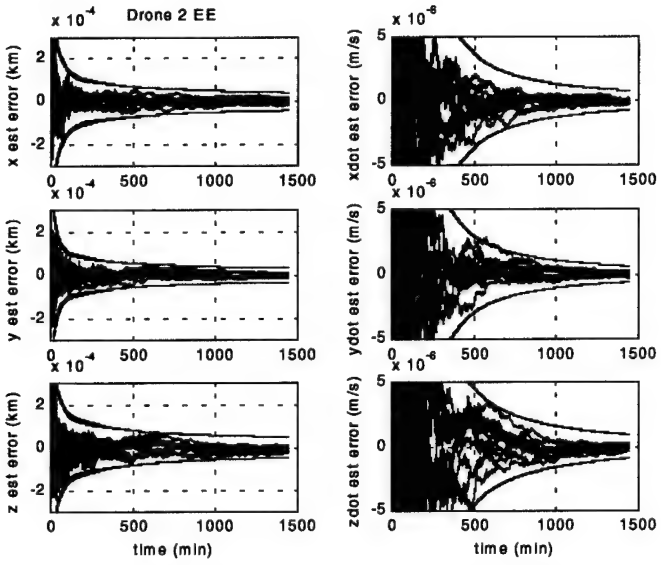

Figure 37. Drone 2 estimation errors (0.5 km focal length)

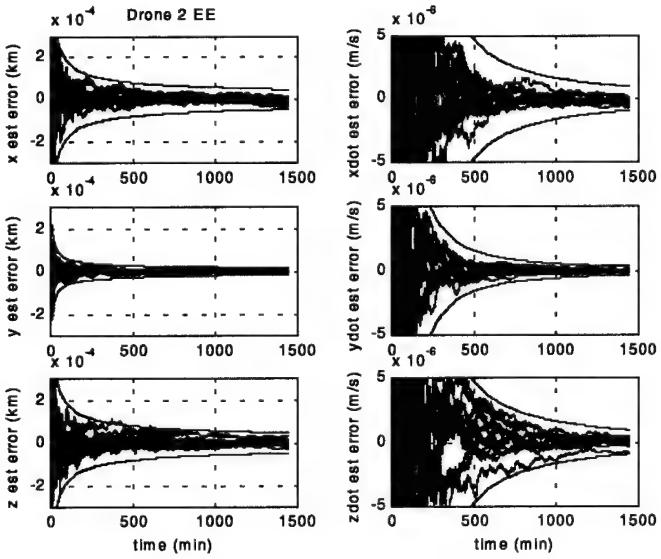

Figure 38. Drone 2 estimation errors (4 km focal length)

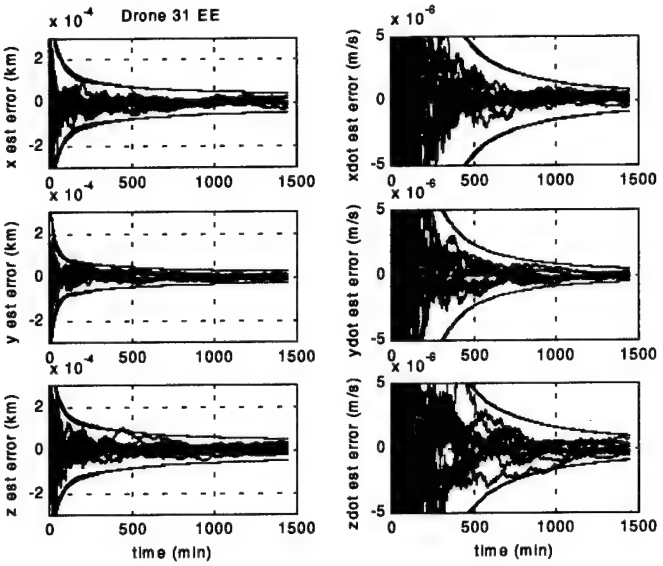

Figure 39. Drone 31 estimation errors (0.5 km focal length)

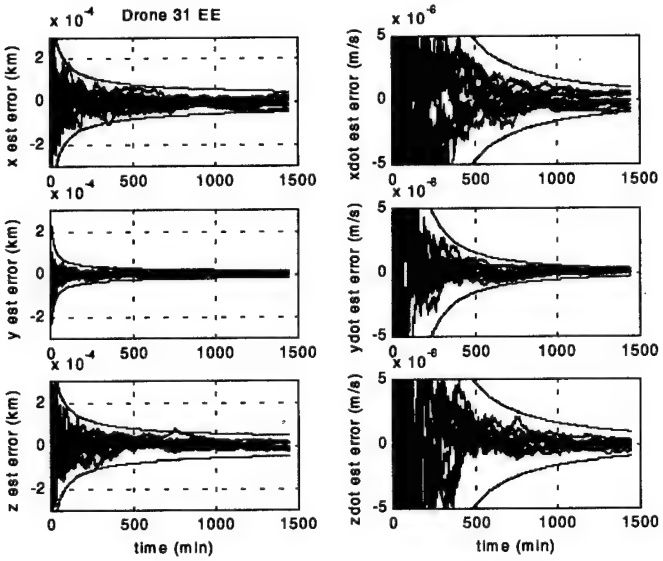

Figure 40. Drone 31 estimation errors (4 km focal length)

For any drone and either focal length, the steady-state position three-sigma values are less than 0.1 meters, and the steady-state velocity three-sigma values are less than 1e-6 meters per second. Also, the estimation errors stay within the three-sigma values with rare exceptions.

From Equations 8.42-8.44 and Equation 8.17, the ΔV can be determined, for each satellite, from the control effort required to reorient the formation. Table 12 gives the average ΔV's from a dozen simulations to reorient the formation.

Focal Length (km)	Hub ΔV (m/s)	Drone 2 ΔV (m/s)	Drone 31 ΔV (m/s)
0.5	1.0126	0.8421	0.8095
4	1.0133	0.8496	0.8190

Table 12. Average formation reorientation ΔV's

The focal length has no discernible effect on the ΔV needed to reorient the formation. This makes sense because the rotation is about the y axis, and the focal length is assumed to be the measurement along the y axis from the hub to the drones. Table 13 gives the propellant masses that correspond to Table 12. Once again, the hub is assumed to have an initial mass of 550 kg, the drones have initial masses of 100 kg, and the Isp's of the ideal thrusters are 10000 seconds.

Focal Length (km)	Hub m_{prop} (g)	Drone 2 m_{prop} (g)	Drone 31 m_{prop} (g)
0.5	5.6771	0.8584	0.8252
4	5.6811	0.8661	0.8349

Table 13. Average formation reorientation propellant masses

Without noise, the ΔV needed to reorient the formation is much less, as shown in Table 14.

Focal Length (km)	Hub ΔV (m/s)	Drone 2 ΔV (m/s)	Drone 31 ΔV (m/s)
0.5	0.0408	0.1529	0.1496
4	0.0408	0.1529	0.1495

Table 14. Required ΔV for formation reorientation without noise

Table 15 gives the propellant masses that correspond to Table 14.

Focal Length (km)	Hub m_{prop} (g)	Drone 2 m_{prop} (g)	Drone 31 m_{prop} (g)
0.5	0.2287	0.1623	0.1525
4	0.2287	0.1623	0.1524

Table 15. Required propellant masses for formation reorientation without noise

9. CONCLUSION

In Section 2, the equations of motion for the circular restricted-three body problem were investigated, and the libration point concept was addressed. By reducing the circular restricted three-body problem to collinear libration points, and linearizing the equations of motion, the problem was simplified enough to apply it to a linear-quadratic-regulator control scheme.

In Section 3, the optimal control was found with the linear-quadratic-regulator method. By making a judicious infinite horizon assumption, the control gain becomes constant. In Section 4, the continuous system was sampled to model a more realistic discrete system, and the discrete optimal control law was found. In Section 5, the system was augmented for multiple satellites, and the fundamentals of formation flying explained.

In Section 6, an observer was introduced to estimate the states of a system by using incoming measurements. The optimal method of updating the estimates was given, to reduce the unwanted process and measurement noise. Hence, the Kalman filter was derived. The Kalman filter was adapted to accommodate nonlinear measurements, with special emphasis on the range, azimuth, and elevation measurements common to spaceflight.

Section 7 gave a background to the Stellar Imager mission, and its performance and design requirements. The preliminary formation, orbit, and control designs were explored in detail. Finally, in Section 8, a reference orbit and associated dynamics specific to the sun-earth L2 point were determined. A more realistic orbit and associated dynamics were found with the use of Generator. Three simulations were developed to

analyze various aspects of the Stellar Imager mission. Using the control and estimation methods developed throughout the thesis, the simulations revealed preliminary tracking errors, estimation errors, and required control efforts for all satellites in the formation. From the control efforts, the necessary ΔV's and propellant masses for orbit maintenance, formation slewing, and formation reorientation were determined.

The control strategy and Kalman filter provided satisfactory results. The hub satellite tracks to its reference orbit sufficiently for the SI mission requirements. The drone satellites, on the other hand, track to only within a few meters. Without noise, though, the drones track to within several micrometers. The first tier of the proposed control scheme for SI requires the drones to track within centimeters. This could be accomplished with better sensors to lessen the effect of the process and measurement noise. Tuning the controller and varying the maneuver intervals should provide additional savings as well. The Kalman filter performed such that the estimation errors were for the most part within the three-sigma values. The propellant mass and ΔV results provide a minimum design boundary for the SI mission. Additional propellant will be needed to perform all attitude maneuvers, tighter control requirement adjustments, and other mission functions.

In the future, many research opportunities exist for formation flying satellite control, libration point dynamics, and the Stellar Imager and other similar missions. Specifically for SI, the attitude dynamics and control problem must be examined and integrated into the translation control problem explored in this thesis. Also, non-ideal thrusters must be modeled as well. In simulation, an implicit assumption is made that the thrusters give perfect, impulsive accelerations determined by the control law.

Realistically, thrusters are misaligned, non-impulsive, and have upper and lower bounds on force outputs. Collision avoidance is another problem. In the simulations, the drones are imaginary points that may pass through each other. In actuality, collisions would cause catastrophic satellite failures. System reliability and determination must be accomplished as well. How a drone failure is detected and how the formation responds to such failures are questions that must be addresed. Nonlinear control and estimation approaches should be considered also. For the nanometer level preciseness of the second and third control tiers, unique control strategies and algorithms must be developed and tested.

REFERENCES

1. Szebehely, V., *Theory of Orbits: The Restricted Problem of Three Bodies.* Academic Press, Inc. 1967.

2. Barden, B.T., and Howell, K.C., "Fundamental Motions Near Collinear Libration Points and Their Transitions," *Journal of the Astronautical Sciences*, Vol. 46. 1998.

3. Howell, K.C., Barden, B.T., and Lo, M.W., "Application of Dynamical Systems Theory to Trajectory Design for a Libration Point Mission," *Journal of the Astronautical Sciences*, Vol. 45. 1997.

4. Farquhar, R., *The Control and Use of Libration-Point Satellites.* NASA Technical Report R-346. 1970.

5. Gomez, G., Masdemont, J., and Simo, C., "Quasihalo Orbits Associated with Libration Points," *Journal of the Astronautical Sciences*, Vol. 46. 1998.

6. Scheeres, D.J., and Vinh, N.X., "Dynamics and Control of Relative Motion in an Unstable Orbit," AIAA Paper 2000-4135. 2000.

7. Hoffman, D.A., *Station-keeping at the Collinear Equilibrium Points of the Earth-Moon System*, JSC-26189. NASA Johnson Space Center. 1993.

8. "Microwave Anisotropy Probe," http://map.gsfc.nasa.gov. October, 2001.

9. "The GSFC Stellar Imager Homepage," http://hires.gsfc.nasa.gov/~si/. October, 2001.

10. "Micro Arcsecond X-Ray Imaging Mission," http://maxim.gsfc.nasa.gov. October, 2001.

11. "MAXIM Pathfinder," http://maxim.gsfc.nasa.gov/pathfinder.html. October, 2001.

12. "Terrestrial Planet Finder," http://tpf.jpl.nasa.gov. October, 2001.

13. Speyer, J.L., "Computation and Transmission Requirements for a Decentralized Linear-Quadratic-Gaussian Control Problem." *IEEE Transactions on Automatic Control*; AC-24(2). 1979.

14. Carpenter, J.R., "A Preliminary Investigation of Decentralized Control for Satellite Formations." Proceedings of the 2000 IEEE Aerospace Conference, March 18-25, 2000.

15. Carpenter, J.R., "Decentralized Control of Satellite Formations." To appear in *International Journal of Robust and Nonlinear Control.*

16. Carpenter, J.R., Folta, D.C., and Quinn, D.A., "Integration of Decentralized Linear-Quadratic-Gaussian Control into GSFC's Universal 3-D Autonomous Formation Flying Algorithm." AIAA Paper 99-4269, AIAA Guidance Navigation &Control, Modeling & Simulation Technologies and Atmospheric Flight Mechanics Conference and Exhibit, August 9-11, 1999.

17. "Magnetospheric Constellation," http://stp.gsfc.nasa.gov/missions/mc/mc.htm. October 2001.

18. "Laser Interferometer Space Antenna," http://lisa.jpl.nasa.gov. October 2001.

19. Wie, B., *Space Vehicle Dynamics and Control.* AIAA Education Series. 1998.

20. Nise, N., *Control Systems Engineering.* 2nd Edition. Addison-Wesley Publishing Company. 1995.

21. Friedland, B., *Control System Design.* McGraw-Hill. 1986.

22. Stengel, R.F., *Optimal Control and Estimation.* Dover Publications, Inc. 1994.

23. Phillips, C.L., and Nagle, H.T., *Digital Control System Analysis and Design.* 3rd Edition. Prentice Hall, Inc. 1995.

24. Bryson, A.E, Jr., and Ho, Y.C., *Applied Optimal Control: Optimization, Estimation, and Control.* Hemisphere Publishing Corporation. 1975.

25. Wagner, C., "Formation Flying Around Sun-Earth Libration Point L1." International Space University. 2000.

26. Brown, R.C., and Hwang, P.Y.C., *Introduction to Random Signals and Applied Kalman Filtering.* 2nd Edition. John Wiley & Sons, Inc. 1992.

27. Maybeck, P.S., *Stochastic Models, Estimation, and Control.* Volume 1. Academic Press, Inc. 1979.

28. Carpenter, K.G., Neff, S.G., Schrijver, C.J., Allen, R.J., and Rajagopal, J., "The Stellar Imager (SI) Mission Concept." In proceedings of the 36th Liege Astrophysical Colloquium: From Optical to Millimetric Intefereometry: Scientific and Technical Challenges. 2001.

29. Carpenter, K.G., Lyon, R.G., Schrijver, C.J., Mundy, L.J., Allen, R.J., and Rajagopal, J., "Imaging the Surfaces and Interiors of Other Stars: The Stellar Imager (SI) Mission Concept." 2001.

30. Rarogiewicz, "Introduction to Interferometry." http://www.mtwilson.edu/Education/Presentations/Interferometry/. October 2001.

31. Golomb, S.W., and Taylor, H. IEEE Trans. Info. Theo., 28, #4. 1982.

32. Leitner, J., and Schnurr, R., "Stellar Imager (SI): Formation Flying." Presentation from the Integrated Mission Design Center, Goddard Space Flight Center, NASA. 2001.

33. US Government Printing Office, *The Astronomical Almanac for the Year 2000.* 1998.

34. Howell, K.C., and Anderson, J., *Generator User's Guide.* Version 3.0.2. 2001.

35. Anderson, J., "Earth-to-Lissajous Transfers and Recovery Trajectories Using an Ephemeris Model." Purdue University. 2001.

36. Humble, R.W., Henry, G.N., and Wiley, J.L., *Space Propulsion Analysis and Design.* The McGraw-Hill Companies, Inc. 1995.

37. Asato, D., "Stellar Imager: Propulsion." Presentation from the Integrated Mission Design Center, Goddard Space Flight Center, NASA. 2001.

Appendix A: Generator-to-Matlab Reference Orbit Conversion Script

getref.m

```
disp('loading reference');
posbig=textread('pos.txt','%s','delimiter','\n');
%pos.txt is Generator output
velbig=textread('vel.txt','%s','delimiter','\n');
%vel.txt is Generator output
for i=1:359,
   for j=1:3,
      p(j)=str2num(posbig{(i-1)*3+j});
      v(j)=str2num(velbig{(i-1)*3+j});
   end
   v=v*86.4;
   stater{i}=[p';v'];
end
disp('reference loaded');
clear posbig velbig p v i j
```

Appendix B: Generator-to-Matlab Dynamics Conversion Script

getA.m

```
disp('loading dynamics');
Abig=textread('Amatrices2.txt','%s','delimiter','\n');
%Amatrices2.txt is Generator output
for i=1:359,
   for j=1:36,
      a(j)=str2num(Abig{(i-1)*87+51+j});
   end
   Agen{i}=reshape(a,6,6)';
end
disp('dynamics loaded');
clear Abig a i j
```

Appendix C: Lissajous Orbit Matlab Simulation

L2orbit.m

```
clear all
clc
close all

%NUMBER OF SATELLITES
k=1;

%MANEUVER INTERVAL
T=1; %(days)

%NUMBER OF YEARS
years=359/365;

%SYSTEM DYNAMICS AND COST WEIGHTING

getA;  %load dynamics from generator output file
getref;%load reference orbit from generator output file

%Define Initial Conditions
for i = 1:k,

  x0(i)=1457251.46633160; %pos in km
  y0(i)=572731.248521048;
  z0(i)=71916.8351826099;
  xdot0(i)=78.4720643736218*86400/1000;
  %vel in km/day (converted fro m/s from generator)
  ydot0(i)=-65.9910673865147*86400/1000;
  zdot0(i)=0.920743113792329*86400/1000;

  if i==1,
     ICj{i}=[x0(i) y0(i) z0(i) xdot0(i) ydot0(i) zdot0(i)]';
     IC0=[ICj{i}];
  else,
```

```
      ICj{i}=[0 0 0 0 0 0]';
      IC0=[IC0;ICj{i}];
  end
end

IC=IC0;

count=0;
for epoch=1:T:365*years,

   if epoch > 1*count,
      count=count+1;
      %BUILD LQR MATRICES
      disp('integrating');
      syms tt ti

      %generator
      Aj(:,:)=Agen{count}(:,:);%time in seconds
      Aj(1:3,1:3)=Aj(1:3,1:3)*86400;%convert to days
      Aj(4:6,1:3)=Aj(4:6,1:3)*86400^2;
      Aj(4:6,4:6)=Aj(4:6,4:6)*86400;

      Bj=[zeros(3);eye(3)];

      Phij1=eye(6)+Aj*ti+Aj^2*ti^2/2;
      Bdj=Phij1*Bj;
      Bdj=int(Bdj,ti,0,tt);

      Phij=eye(6)+Aj*T+Aj^2*T^2/2;

      Wj=[eye(3) zeros(3);zeros(3) eye(3)*.001]*1e6;
      Vj=eye(3);

      Wdj=transpose(Phij)*Wj*(Phij);
      Wdj=int(Wdj,tt,0,T);

      Vdj=(transpose(Bdj)*Wj*Bdj+Vj);
      Vdj=int(Vdj,tt,0,T);
```

```
Mdj=transpose(Phij)*Wj*Bdj;
Mdj=int(Mdj,tt,0,T);

%PROCESS NOISE
qnode=1e-6;
Qj=[zeros(3) zeros(3);zeros(3) eye(3)*qnode];
Qdj=int(Phij1*Qj*transpose(Phij1),ti,0,T);

tt=T;

%EVALUATE DISCRETE PARTS
Adj=Phij;%state transition matrix
Bdj=eval(Bdj); %discrete control mapping matrix
Wdj=eval(Wdj); %discrete state weighting matrix
Vdj=eval(Vdj); %discrete control weighting matrix
Mdj=eval(Mdj); %discrete state-control couple weight
Qdj=eval(Qdj); %discrete process noise matrix

%APPEND FOR MULTIPLE SPACECRAFT
for j=k:-1:1,
   A(6*j-5:6*j,6*j-5:6*j)=Aj;
   Ad(6*j-5:6*j,6*j-5:6*j)=Adj;
   W(6*j-5:6*j,6*j-5:6*j)=Wj;
   Wd(6*j-5:6*j,6*j-5:6*j)=Wdj;
   V(3*j-2:3*j,3*j-2:3*j)=Vj;
   Vd(3*j-2:3*j,3*j-2:3*j)=Vdj;
   Md(6*j-5:6*j,3*j-2:3*j)=Mdj;
   Q(6*j-5:6*j,6*j-5:6*j)=Qj;
   Qd(6*j-5:6*j,6*j-5:6*j)=Qdj;
   if j==1,
      Bd(6*j-5:6*j,3*j-2:3*j)=Bdj;
      B(6*j-5:6*j,3*j-2:3*j)=Bj;
   else,
      Bd(6*j-5:6*j,3*j-2:3*j)=Bdj;
      Bd(6*j-5:6*j,1:3)=-1*Bdj;
      B(6*j-5:6*j,3*j-2:3*j)=Bj;
      B(6*j-5:6*j,1:3)=-1*Bj;
```

```
      end
   end

   %MEASUREMENTS
   R1=diag([.1 .1*3/1500000 .1*3/1500000].^2);
   Rj=diag([.0001 .0001*3/(1/2) .0001*3/(1/2)].^2);
   R(1:3,1:3)=R1;
   for j=k:-1:2,
      R(3*j-2:3*j,3*j-2:3*j)=Rj;
   end

   Pnot1=diag([1 1 1 86.4 86.4 86.4].^2);
   Pnotj=diag([.001 .001 .001 .0864 .0864 .0864].^2);
   Pnot(1:6,1:6)=Pnot1;
   for j=k:-1:2,
      Pnot(6*j-5:6*j,6*j-5:6*j)=Pnotj;
   end

   disp('calculating gain');
   Kd=dlqr(Ad,Bd,Wd,Vd,Md); %discrete control gain
end

%SIMULATION AROUND L2
if epoch == 1,
   state{epoch}=IC; %start at initial conditions
   stater{epoch}=[stater{epoch};IC(7:end)]; %initialize reference for system
   statedif{epoch}=state{epoch}-stater{epoch}; %calculate state error
   stateest{epoch}=statedif{epoch}; %initialize error estimate
   Ppos=Pnot;
   for j=k:-1:1,
      sigmax{epoch}(j)=sqrt(Ppos(6*j-5,6*j-5));
      sigmay{epoch}(j)=sqrt(Ppos(6*j-4,6*j-4));
      sigmaz{epoch}(j)=sqrt(Ppos(6*j-3,6*j-3));
      sigmaxdot{epoch}(j)=sqrt(Ppos(6*j-2,6*j-2));
      sigmaydot{epoch}(j)=sqrt(Ppos(6*j-1,6*j-1));
      sigmazdot{epoch}(j)=sqrt(Ppos(6*j,6*j));
   end
   epoch
```

```
else
   w=sqrt(Q/T)*randn(length(Q),1); %random process noise
   u{epoch-T}=-Kd*(stateest{epoch-T});
   statedif{epoch}=Ad*statedif{epoch-T}+Bd*u{epoch-T}+w;
   stateest{epoch}=Ad*stateest{epoch-T}+Bd*u{epoch-T};

   if k > 1,   %drone references don't change
      for el=2:k,
         stater{epoch}(6*el-5:6*el)=stater{epoch-T}(6*el-5:6*el);
      end
   end

   statebar{epoch}=stateest{epoch}+stater{epoch};
   state{epoch}=statedif{epoch}+stater{epoch};

   v=chol(R)'*randn(length(R),1); %random measurement noise

   %calculate measurements (true and estimated)
   rtrue=[];
   rbar=[];
   for el=1:k,
      postrue{el}=state{epoch}(6*el-5:6*el-3);
      rangetrue(el)=(transpose(postrue{el})*postrue{el})^.5;
      if rangetrue(el)==0,
         elevtrue(el)=0;
         aztrue(el)=0;
      else
         elevtrue(el)=asin(-postrue{el}(3)/rangetrue(el));
         aztrue(el)=asin(postrue{el}(1)/rangetrue(el)/cos(elevtrue(el)));
      end

      pos{el}=statebar{epoch}(6*el-5:6*el-3);
      range(el)=(transpose(pos{el})*pos{el})^.5;
      if range(el)==0,
         elev(el)=0;
         az(el)=0;
      else
         elev(el)=asin(-pos{el}(3)/range(el));
```

```
            az(el)=asin(pos{el}(1)/range(el)/cos(elev(el)));
         end

         rtrue=[rtrue;rangetrue(el);elevtrue(el);aztrue(el)];
         rbar=[rbar;range(el);elev(el);az(el)];
      end

   Y{epoch}=rtrue+v;  %actual measurement (based on state)

   %calculate H
   H=[];
   for el=k:-1:1,
      if range(el)==0,
         Ha=[zeros(3,6)];
      elseif pos{el}(1)^2==range(el)^2*cos(elev(el))^2,
         Ha=[transpose(pos{el})/range(el) 0 0 0;...
               pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               pos{el}(2)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               -(1/range(el)-pos{el}(3)^2/range(el)^3)/cos(elev(el)) 0 0 0;...
               0 0 0 0 0 0];
      elseif cos(elev(el))==0,
         Ha=[transpose(pos{el})/range(el) 0 0 0;zeros(2,6)];
      else
         Ha=[transpose(pos{el})/range(el) 0 0 0;...
               pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               pos{el}(2)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               -(1/range(el)-pos{el}(3)^2/range(el)^3)/cos(elev(el)) 0 0 0;...
               (1/range(el)/cos(elev(el))-pos{el}(1)^2/range(el)^3/cos(elev(el))-
pos{el}(1)^2*pos{el}(3)^2/range(el)^5/cos(elev(el))^3)/...
               (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 ...
               (-pos{el}(1)*pos{el}(2)/range(el)^3/cos(elev(el))-
pos{el}(1)*pos{el}(2)*pos{el}(3)^2/range(el)^5/cos(elev(el))^3)/...
               (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 ...
               (-pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el))-.5*pos{el}(1)*(-
2*pos{el}(3)/range(el)^2+2*pos{el}(3)^3/range(el)^4)/range(el)/cos(elev(el))^3)/...
               (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 0 0 0];
      end
      H(3*el-2:3*el,6*el-5:6*el)=Ha;
```

```
   end

   Pneg=Ad*Ppos*transpose(Ad)+Qd; %propagate error covariance
   L=Pneg*transpose(H)*inv(R+H*Pneg*transpose(H)); %calculate Kalman gain
   stateest{epoch}=stateest{epoch}+L*(Y{epoch}-rbar); %update error estimate
   Ppos=Pneg-L*H*Pneg; %update error covariance

   for j=k:-1:1,
      sigmax{epoch}(j)=sqrt(Ppos(6*j-5,6*j-5));
      sigmay{epoch}(j)=sqrt(Ppos(6*j-4,6*j-4));
      sigmaz{epoch}(j)=sqrt(Ppos(6*j-3,6*j-3));
      sigmaxdot{epoch}(j)=sqrt(Ppos(6*j-2,6*j-2));
      sigmaydot{epoch}(j)=sqrt(Ppos(6*j-1,6*j-1));
      sigmazdot{epoch}(j)=sqrt(Ppos(6*j,6*j));
   end
   epoch
end
end

u{epoch}=-Kd*[zeros(1,6*k)]';

t=1:T:epoch;
%Convert u, state, and stater to matrices for plotting
for i=1:T:epoch,
   for j=1:3*k,
      um(i,j)=u{i}(j);
   end
   for j=1:6*k,
      statem(i,j)=state{i}(j);
      staterm(i,j)=stater{i}(j);
      statedifm(i,j)=statedif{i}(j);
      stateestm(i,j)=stateest{i}(j);
   end
   for j=1:k,
      sigmaxm(i,j)=sigmax{i}(j);
      sigmaym(i,j)=sigmay{i}(j);
      sigmazm(i,j)=sigmaz{i}(j);
      sigmaxdotm(i,j)=sigmaxdot{i}(j);
```

```
      sigmaydotm(i,j)=sigmaydot{i}(j);
      sigmazdotm(i,j)=sigmazdot{i}(j);
   end
end
%Need to make vectors same length for plotting when T /= 1
for i=1:T:epoch,
   for el=1:T-1,
      statem(i+el,:)=statem(i,:);
      staterm(i+el,:)=staterm(i,:);
      statedifm(i+el,:)=statedifm(i,:);
      stateestm(i+el,:)=stateestm(i,:);
      um(i+el,:)=um(i,:)*0;%No control when a maneuver not performed
   end
end

%Total Delta V
for i=1:k,
   dvx(i)=sum(abs(um(:,3*i-2)));
   dvy(i)=sum(abs(um(:,3*i-1)));
   dvz(i)=sum(abs(um(:,3*i)));
   totdv(i)=dvx(i)+dvy(i)+dvz(i);  %km/day^2
end
totdv1=totdv*T;       %km/day
totdv1=totdv1*1000/86400   %m/s
disp('in m/s');

mi1=550;
mi2=100;
Isp1=10000;
Isp2=10000;
massprop(1)=mi1*(1-exp(-totdv1(1)/Isp1/9.81));
for i=2:k,
   massprop(i)=mi2*(1-exp(-totdv1(i)/Isp2/9.81));
end
massprop  %kg
disp('in kg');

%PLOTS
```

```
color=['b','r','g','m','c','y'];
color=[color,color,color,color,color,color];

if k > 1,

%hub satellite truth
figure;
subplot(2,2,4),plot3(statem(:,1),statem(:,2),statem(:,3),'b',0,0,0,'*k',statem(1,1),statem(1,2),statem(1,3)
,'or');grid on;
xlabel('x (km)');ylabel('y (km)');zlabel('z (km)');
subplot(2,2,1),plot(statem(:,1),statem(:,2),'b',0,0,'*k',statem(1,1), ...
   statem(1,2),'or');xlabel('x (km)');ylabel('y (km)');grid on;title('hub true position');
subplot(2,2,2),plot(statem(:,1),statem(:,3),'b',0,0,'*k',statem(1,1), ...
   statem(1,3),'or');xlabel('x (km)');ylabel('z (km)');grid on;
subplot(2,2,3),plot(statem(:,2),statem(:,3),'b',0,0,'*k',statem(1,2), ...
   statem(1,3),'or');xlabel('y (km)');ylabel('z (km)');grid on;

%hub satellite reference
figure;
subplot(2,2,4),plot3(staterm(:,1),staterm(:,2),staterm(:,3),'b',0,0,0,'*k');grid on;
xlabel('x (km)');ylabel('y (km)');zlabel('z (km)');
subplot(2,2,1),plot(staterm(:,1),staterm(:,2),'b',0,0,'*k');xlabel('x (km)');ylabel('y (km)');grid
on;title('reference for hub satellite');
subplot(2,2,2),plot(staterm(:,1),staterm(:,3),'b',0,0,'*k');xlabel('x (km)');ylabel('z (km)');grid on;
subplot(2,2,3),plot(staterm(:,2),staterm(:,3),'b',0,0,'*k');xlabel('y (km)');ylabel('z (km)');grid on;

%hub tracking errors
figure;
subplot(3,2,1),plot(t,statedifm(:,1));ylabel('x error');grid on;
title('hub tracking error');
subplot(3,2,3),plot(t,statedifm(:,2));ylabel('y error');grid on;
subplot(3,2,5),plot(t,statedifm(:,3));ylabel('z error');grid on;xlabel('time');
subplot(3,2,2),plot(t,statedifm(:,4));ylabel('xdot error');grid on;
subplot(3,2,4),plot(t,statedifm(:,5));ylabel('ydot error');grid on;
subplot(3,2,6),plot(t,statedifm(:,6));ylabel('zdot error');grid on;xlabel('time');

%hub estimation errors
figure;
```

```
subplot(3,2,1),plot(t,statedifm(:,1)-stateestm(:,1),t,3*sigmaxm(:,1),'r',t,-3*sigmaxm(:,1),'r');
ylabel('x estimation error (km)');grid on;title('hub estimation error');
subplot(3,2,3),plot(t,statedifm(:,2)-stateestm(:,2),t,3*sigmaym(:,1),'r',t,-3*sigmaym(:,1),'r');
ylabel('y estimation error (km)');grid on;
subplot(3,2,5),plot(t,statedifm(:,3)-stateestm(:,3),t,3*sigmazm(:,1),'r',t,-3*sigmazm(:,1),'r');
ylabel('z estimation error (km)');xlabel('time');grid on;
subplot(3,2,2),plot(t,(statedifm(:,4)-stateestm(:,4))*1000/86400,t,3*sigmaxdotm(:,1)*1000/86400,'r',t,-
3*sigmaxdotm(:,1)*1000/86400,'r');
ylabel('xdot estimation error (m/s)');grid on;
subplot(3,2,4),plot(t,(statedifm(:,5)-stateestm(:,5))*1000/86400,t,3*sigmaydotm(:,1)*1000/86400,'r',t,-
3*sigmaydotm(:,1)*1000/86400,'r');
ylabel('ydot estimation error (m/s)');grid on;
subplot(3,2,6),plot(t,(statedifm(:,6)-stateestm(:,6))*1000/86400,t,3*sigmazdotm(:,1)*1000/86400,'r',t,-
3*sigmazdotm(:,1)*1000/86400,'r');
ylabel('zdot estimation error (m/s)');xlabel('time');grid on;

%hub satellite controls
figure;
subplot(3,1,1),plot(t,um(:,1));ylabel('control x');grid on;
title(['Control ',int2str(1)]);
subplot(3,1,2),plot(t,um(:,2));ylabel('control y');grid on;
subplot(3,1,3),plot(t,um(:,3));ylabel('control z');xlabel('time');grid on;

for i=2:k,
   %relative satellite positions
   figure;
   subplot(2,2,4),plot3(statem(:,6*i-5),statem(:,6*i-4),statem(:,6*i-3),color(i),0,0,0,'*k');grid on;
   xlabel('x');ylabel('y');zlabel('z');title(['relative satellite ',int2str(i)]);
   subplot(2,2,1),plot(statem(:,6*i-5),statem(:,6*i-4),color(i),0,0,'*b');xlabel('x');ylabel('y');grid on;
   subplot(2,2,2),plot(statem(:,6*i-5),statem(:,6*i-3),color(i),0,0,'*b');xlabel('x');ylabel('z');grid on;
   subplot(2,2,3),plot(statem(:,6*i-4),statem(:,6*i-3),color(i),0,0,'*b');xlabel('y');ylabel('z');grid on;

   %relative satellite references
   figure;
   subplot(2,2,4),plot3(staterm(:,6*i-5),staterm(:,6*i-4),staterm(:,6*i-3),color(i),0,0,0,'*b');grid on;
   xlabel('x');ylabel('y');zlabel('z');title(['relative reference ',int2str(i)]);
   subplot(2,2,1),plot(staterm(:,6*i-5),staterm(:,6*i-4),color(i),0,0,'*b');xlabel('x');ylabel('y');grid
on;
```

```
   subplot(2,2,2),plot(staterm(:,6*i-5),staterm(:,6*i-3),color(i),0,0,'*b');xlabel('x');ylabel('z');grid
on;
   subplot(2,2,3),plot(staterm(:,6*i-4),staterm(:,6*i-3),color(i),0,0,'*b');xlabel('y');ylabel('z');grid
on;

   %relative satellite tracking errors
   figure;
   subplot(3,2,1),plot(t,statedifm(:,6*i-5));ylabel('x error');grid on;
   title(['drone ' num2str(i) ' tracking error']);
   subplot(3,2,3),plot(t,statedifm(:,6*i-4));ylabel('y error');grid on;
   subplot(3,2,5),plot(t,statedifm(:,6*i3));ylabel('z error');grid on;xlabel('time');
   subplot(3,2,2),plot(t,statedifm(:,6*i-2));ylabel('xdot error');grid on;
   subplot(3,2,4),plot(t,statedifm(:,6*i-1));ylabel('ydot error');grid on;
   subplot(3,2,6),plot(t,statedifm(:,6*i));ylabel('zdot error');grid on;xlabel('time');

   %relative satellite estimation errors
   figure;
   subplot(3,2,1),plot(t,statedifm(:,6*i-5)-stateestm(:,6*i-5),t,3*sigmaxm(:,i),'r',t,-3*sigmaxm(:,i),'r');
   ylabel('x estimation error (km)');grid on;title(['drone ' num2str(i) ' estimation error']);
   subplot(3,2,3),plot(t,statedifm(:,6*i-4)-stateestm(:,6*i-4),t,3*sigmaym(:,i),'r',t,-3*sigmaym(:,i),'r');
   ylabel('y estimation error (km)');grid on;
   subplot(3,2,5),plot(t,statedifm(:,6*i-3)-stateestm(:,6*i-3),t,3*sigmazm(:,i),'r',t,-3*sigmazm(:,i),'r');
   ylabel('z estimation error (km)');xlabel('time');grid on;
   subplot(3,2,2),plot(t,(statedifm(:,6*i-2)-stateestm(:,6*i-
2))*1000/86400,t,3*sigmaxdotm(:,i)*1000/86400,'r',t,-3*sigmaxdotm(:,i)*1000/86400,'r');
   ylabel('xdot estimation error (m/s)');grid on;
   subplot(3,2,4),plot(t,(statedifm(:,6*i-1)-stateestm(:,6*i-
1))*1000/86400,t,3*sigmaydotm(:,i)*1000/86400,'r',t,-3*sigmaydotm(:,i)*1000/86400,'r');
   ylabel('ydot estimation error (m/s)');grid on;
   subplot(3,2,6),plot(t,(statedifm(:,6*i)-
stateestm(:,6*i))*1000/86400,t,3*sigmazdotm(:,i)*1000/86400,'r',t,-3*sigmazdotm(:,i)*1000/86400,'r');
   ylabel('zdot estimation error (m/s)');xlabel('time');grid on;

   %relative satellite controls
   figure;
   subplot(3,1,1),plot(t,um(:,3*i-2),color(i));ylabel('control x');grid on;
   title(['relative control ',int2str(i)]);
   subplot(3,1,2),plot(t,um(:,3*i-1),color(i));ylabel('control y');grid on;
```

```
   subplot(3,1,3),plot(t,um(:,3*i),color(i));ylabel('control z');xlabel('time');grid on;
end
else,
   %hub satellite truth
   figure;
   subplot(2,2,4),
   plot3(statem(:,1),statem(:,2),statem(:,3),'b',0,0,0,'*k',statem(1,1),statem(1,2),statem(1,3),'or');grid
on;hold off;
   xlabel('x');ylabel('y');zlabel('z');title('true positions');

   subplot(2,2,1),
   plot(statem(:,1),statem(:,2),'b',0,0,'*k',statem(1,1),statem(1,2),'or');xlabel('x');ylabel('y');grid on;
   hold off;

   subplot(2,2,2),
plot(statem(:,1),statem(:,3),'b',0,0,'*k',statem(1,1),statem(1,3),'or');xlabel('x');ylabel('z');grid on;
hold off;

subplot(2,2,3),
plot(statem(:,2),statem(:,3),'b',0,0,'*k',statem(1,2),statem(1,3),'or');xlabel('y');ylabel('z');grid on;
hold off;

%hub satellite reference
figure;
subplot(2,2,4),plot3(staterm(:,1),staterm(:,2),staterm(:,3),'b',0,0,0,'*k',staterm(1,1),staterm(1,2),stater
m(1,3),'or');grid on;
xlabel('x');ylabel('y');zlabel('z');title('reference for hub satellite');
subplot(2,2,1),plot(staterm(:,1),staterm(:,2),'b',0,0,'*k',staterm(1,1),staterm(1,2),'or');xlabel('x');ylab
el('y');grid on;
subplot(2,2,2),plot(staterm(:,1),staterm(:,3),'b',0,0,'*k',staterm(1,1),staterm(1,3),'or');xlabel('x');ylab
el('z');grid on;
subplot(2,2,3),plot(staterm(:,2),staterm(:,3),'b',0,0,'*k',staterm(1,2),staterm(1,3),'or');xlabel('y');ylab
el('z');grid on;

%hub satellite tracking errors
figure;
subplot(3,2,1),plot(t,statedifm(:,1));ylabel('x error (km)');grid on;
title('hub tracking error');
```

```
subplot(3,2,3),plot(t,statedifm(:,2));ylabel('y error (km)');grid on;
subplot(3,2,5),plot(t,statedifm(:,3));ylabel('z error (km)');grid on;xlabel('time');
subplot(3,2,2),plot(t,statedifm(:,4)*1000/86400);ylabel('xdot error (m/s)');grid on;
subplot(3,2,4),plot(t,statedifm(:,5)*1000/86400);ylabel('ydot error (m/s)');grid on;
subplot(3,2,6),plot(t,statedifm(:,6)*1000/86400);ylabel('zdot error (m/s)');grid on;xlabel('time');

%hub estimation errors
figure;
subplot(3,2,1),plot(t,statedifm(:,1)-stateestm(:,1),t,3*sigmaxm(:,1),'r',t,-3*sigmaxm(:,1),'r');
ylabel('x estimation error (km)');grid on;title('hub estimation error');
subplot(3,2,3),plot(t,statedifm(:,2)-stateestm(:,2),t,3*sigmaym(:,1),'r',t,-3*sigmaym(:,1),'r');
ylabel('y estimation error (km)');grid on;
subplot(3,2,5),plot(t,statedifm(:,3)-stateestm(:,3),t,3*sigmazm(:,1),'r',t,-3*sigmazm(:,1),'r');
ylabel('z estimation error (km)');xlabel('time');grid on;
subplot(3,2,2),plot(t,(statedifm(:,4)-stateestm(:,4))*1000/86400,t,3*sigmaxdotm(:,1)*1000/86400,'r',t,-
3*sigmaxdotm(:,1)*1000/86400,'r');
ylabel('xdot estimation error (m/s)');grid on;
subplot(3,2,4),plot(t,(statedifm(:,5)-stateestm(:,5))*1000/86400,t,3*sigmaydotm(:,1)*1000/86400,'r',t,-
3*sigmaydotm(:,1)*1000/86400,'r');
ylabel('ydot estimation error (m/s)');grid on;
subplot(3,2,6),plot(t,(statedifm(:,6)-stateestm(:,6))*1000/86400,t,3*sigmazdotm(:,1)*1000/86400,'r',t,-
3*sigmazdotm(:,1)*1000/86400,'r');
ylabel('zdot estimation error (m/s)');xlabel('time');grid on;

%hub satellite controls
figure;
subplot(3,1,1),plot(t,um(:,1));ylabel('control x');grid on;
title(['Control ',int2str(1)]);
subplot(3,1,2),plot(t,um(:,2));ylabel('control y');grid on;
subplot(3,1,3),plot(t,um(:,3));ylabel('control z');xlabel('time');grid on;

end
```

Appendix D: Formation Slewing Matlab Simulation

Formslew.m

```
%FORMATION SLEWING SIMULATION
clear all
close all

%NUMBER OF SATELLITES
k=31;
%PLACE IN LISSAJOUS ORBIT (1-358)
place=2;
%MANEUVER INTERVAL
dT=1; %(minutes)
%NUMBER OF HOURS TO RUN
hours=24;
%FOCAL LENGTH
fl=0.5; %km (0.5 or 4 for SI)

rad=pi/180;
%SLEW ANGLE (0-360 deg)
slew=90*rad;

getA;
getref;

Aj(:,:)=Agen{place}(:,:);%time in seconds
Aj(1:3,1:3)=Aj(1:3,1:3)*86400;%convert to days
Aj(4:6,1:3)=Aj(4:6,1:3)*86400^2;
Aj(4:6,4:6)=Aj(4:6,4:6)*86400;

statera=stater{place};
staterb=stater{place+1};

syms tt ti
T=1;
%SYSTEM DYNAMICS AND COST WEIGHTING
```

```
disp('integrating');

Bj=[zeros(3);eye(3)];

Phij1=eye(6)+Aj*ti+Aj^2*ti^2/2;
Bdj=Phij1*Bj;
Bdj=int(Bdj,ti,0,tt);

Phij=eye(6)+Aj*dT/1440+Aj^2*(dT/1440)^2/2;
%WEIGHTING MATRICES
Wj=[eye(3) zeros(3);zeros(3) eye(3)*.001]*1e6;
Vj=eye(3);

Wdj=transpose(Phij)*Wj*(Phij);
Wdj=int(Wdj,tt,0,dT/1440);

Vdj=(transpose(Bdj)*Wj*Bdj+Vj);
Vdj=int(Vdj,tt,0,dT/1440);

Mdj=transpose(Phij)*Wj*Bdj;
Mdj=int(Mdj,tt,0,dT/1440);

%PROCESS NOISE
qnode=1e-24;
Qj=[zeros(3) zeros(3);zeros(3) eye(3)*qnode];
Qdj=int(Phij*Qj*transpose(Phij),ti,0,dT/1440);

tt=dT/1440;
%DISCRETE PARTS
Adj=Phij;%state transition matrix
Bdj=eval(Bdj); %discrete control mapping matrix
Wdj=eval(Wdj); %discrete state weighting matrix
Vdj=eval(Vdj); %discrete control weighting matrix
Mdj=eval(Mdj); %disrete state-control couple weight
Qdj=eval(Qdj); %discrete process noise matrix
for j=k:-1:1,
   A(6*j-5:6*j,6*j-5:6*j)=Aj;
   Ad(6*j-5:6*j,6*j-5:6*j)=Adj;
```

```
   Wd(6*j-5:6*j,6*j-5:6*j)=Wdj;
   Vd(3*j-2:3*j,3*j-2:3*j)=Vdj;
   Md(6*j-5:6*j,3*j-2:3*j)=Mdj;
   Qd(6*j-5:6*j,6*j-5:6*j)=qnode*Qdj;
   Q(6*j-5:6*j,6*j-5:6*j)=Qj;
   if j==1,
      Bd(6*j-5:6*j,3*j-2:3*j)=Bdj;
   else,
      Bd(6*j-5:6*j,3*j-2:3*j)=Bdj;
      Bd(6*j-5:6*j,1:3)=-1*Bdj;
   end
end

disp('calculating gain');
Kd=dlqr(Ad,Bd,Wd,Vd,Md); %discrete control gain

%MEASUREMENTS
R1=diag([.1 .1*3/1500000 .1*3/1500000].^2);
Rj=diag([.0001 .0001*3/fl .0001*3/fl].^2);
R(1:3,1:3)=R1;
for j=k:-1:2,
   R(3*j-2:3*j,3*j-2:3*j)=Rj;
end

Pnot1=diag([1 1 1 86.4 86.4 86.4].^2);
Pnotj=diag([.001 .001 .001 .0864 .0864 .0864].^2);
Pnot(1:6,1:6)=Pnot1;
for j=k:-1:2,
   Pnot(6*j-5:6*j,6*j-5:6*j)=Pnotj;
end

%DEFINE initial GEOMETRY (spherical coordinates)
for i=2:k,
   dr(i)=2*fl;%all radial components
end
for i=2:16,
      dphi(i)=(3*pi/2+asin(.25/dr(2))/15*(17-i));
   end
```

```
for i=17:k,
   dphi(i)=(3*pi/2-asin(.25/dr(2))/15*(i-16));
end
dtheta(2)=-asin(.25/dr(2))/15*15;
dtheta(3)=asin(.25/dr(2))/15*10;
dtheta(4)=-asin(.25/dr(2))/15*14;
dtheta(5)=asin(.25/dr(2))/15*4;
dtheta(6)=asin(.25/dr(2))/15*6;
dtheta(7)=asin(.25/dr(2))/15*11;
dtheta(8)=asin(.25/dr(2))/15*14;
dtheta(9)=-asin(.25/dr(2))/15*3;
dtheta(10)=-asin(.25/dr(2))/15*13;
dtheta(11)=-asin(.25/dr(2))/15*1;
dtheta(12)=asin(.25/dr(2))/15*9;
dtheta(13)=asin(.25/dr(2))/15*5;
dtheta(14)=-asin(.25/dr(2))/15*4;
dtheta(15)=asin(.25/dr(2))/15*8;
dtheta(16)=asin(.25/dr(2))/15*7;
dtheta(17)=-asin(.25/dr(2))/15*9;
dtheta(18)=-asin(.25/dr(2))/15*8;
dtheta(19)=asin(.25/dr(2))/15*12;
dtheta(20)=-asin(.25/dr(2))/15*11;
dtheta(21)=-asin(.25/dr(2))/15*7;
dtheta(22)=asin(.25/dr(2))/15*15;
dtheta(23)=asin(.25/dr(2))/15*3;
dtheta(24)=asin(.25/dr(2))/15*13;
dtheta(25)=-asin(.25/dr(2))/15*2;
dtheta(26)=-asin(.25/dr(2))/15*5;
dtheta(27)=-asin(.25/dr(2))/15*10;
dtheta(28)=-asin(.25/dr(2))/15*12;
dtheta(29)=asin(.25/dr(2))/15*2;
dtheta(30)=-asin(.25/dr(2))/15*6;
dtheta(31)=asin(.25/dr(2))/15*1;

dphicent=3*pi/2;
dthetacent=0;

%convert to cartesian
```

```
dycent=dr(2)*sin(dphicent)*cos(dthetacent);
dxcent=dr(2)*sin(dthetacent);
dzcent=dr(2)*cos(dphicent)*cos(dthetacent);

for i=2:k,
   dyj(i)=dr(i)*sin(dphi(i))*cos(dtheta(i));
   dxj(i)=dr(i)*sin(dtheta(i));
   dzj(i)=dr(i)*cos(dphi(i))*cos(dtheta(i));
   dy(i)=dyj(i)-dycent/2;
   dx(i)=dxj(i)-dxcent/2;
   dz(i)=dzj(i)-dzcent/2;
end

IC=[];
for i=1:k,
   if i==1,
      dx(i)=statera(1);
      dy(i)=statera(2);
      dz(i)=statera(3);
      dxdot(i)=statera(4);
      dydot(i)=statera(5);
      dzdot(i)=statera(6);
      IC=[IC;dx(i);dy(i);dz(i);dxdot(i);dydot(i);dzdot(i)];
   else,
      IC=[IC;dx(i);dy(i);dz(i);0;0;0];
   end
end

%Define reference geometry
dx1(1)=statera(1);%hub
dy1(1)=statera(2);
dz1(1)=statera(3);

for i = 2:k,
   dr(i)=2*fl;
   dphi1(i)=dphi(i)-slew;
   dtheta1(i)=dtheta(i);
end
```

```
dthetacent=0;
dphicent=(3*pi/2-slew);

%convert to cartesian
for i=2:k,
   dyj1(i)=dr(i)*sin(dphi1(i))*cos(dtheta1(i));
   dxj1(i)=dr(i)*sin(dtheta1(i));
   dzj1(i)=dr(i)*cos(dphi1(i))*cos(dtheta1(i));
   dycent=dr(i)*sin(dphicent)*cos(dthetacent);
   dxcent=dr(i)*sin(dthetacent);
   dzcent=dr(i)*cos(dphicent)*cos(dthetacent);

   dy1(i)=dyj1(i)-dycent/2;
   dx1(i)=dxj1(i)-dxcent/2;
   dz1(i)=dzj1(i)-dzcent/2;
end


ref1=[];
for i=1:k,
   if i==1,
      dxdot1(i)=statera(4);
      dydot1(i)=statera(5);
      dzdot1(i)=statera(6);
      ref1=[ref1;dx1(i);dy1(i);dz1(i);dxdot1(i);dydot1(i);dzdot1(i)];
   else,
      ref1=[ref1;dx1(i);dy1(i);dz1(i);0;0;0];
   end
end

%SIMULATE
for i=1:dT:60*hours,
   if i == 1,
      state1{i}=IC;
      stater1{i}=ref1;
      statedif1{i}=state1{i}-stater1{i};
      stateest1{i}=statedif1{i};
```

```
    Ppos=Pnot;
    for j=k:-1:1,
       sigmax{i}(j)=sqrt(Ppos(6*j-5,6*j-5));
       sigmay{i}(j)=sqrt(Ppos(6*j-4,6*j-4));
       sigmaz{i}(j)=sqrt(Ppos(6*j-3,6*j-3));
       sigmaxdot{i}(j)=sqrt(Ppos(6*j-2,6*j-2));
       sigmaydot{i}(j)=sqrt(Ppos(6*j-1,6*j-1));
       sigmazdot{i}(j)=sqrt(Ppos(6*j,6*j));
    end
    i
else
    w=sqrt(Q/T)*randn(length(Q),1);
    u1{i-dT}=-Kd*(stateest1{i-dT});

    stater1{i}=stater1{i-dT}; %just for sizing
    stater1{i}(1:6)=(staterb(1:6)-stater1{1}(1:6))/(60*hours/dT)*i+stater1{1}(1:6);

    statedif1{i}=Ad*statedif1{i-dT}+Bd*u1{i-dT}+w;
    stateest1{i}=Ad*stateest1{i-dT}+Bd*u1{i-dT};

    statebar1{i}=stateest1{i}+stater1{i};
    state1{i}=statedif1{i}+stater1{i};

    v=chol(R)'*randn(length(R),1);

    rtrue=[];
    rbar=[];
    for el=1:k,
       postrue{el}=state1{i}(6*el-5:6*el-3);
       rangetrue(el)=(transpose(postrue{el})*postrue{el})^.5;
       elevtrue(el)=asin(-postrue{el}(3)/rangetrue(el));
       aztrue(el)=asin(postrue{el}(1)/rangetrue(el)/cos(elevtrue(el)));

       pos{el}=statebar1{i}(6*el-5:6*el-3);
       range(el)=(transpose(pos{el})*pos{el})^.5;
       elev(el)=asin(-pos{el}(3)/range(el));
       az(el)=asin(pos{el}(1)/range(el)/cos(elev(el)));
```

```
            rtrue=[rtrue;rangetrue(el);elevtrue(el);aztrue(el)];
            rbar=[rbar;range(el);elev(el);az(el)];
         end

         Y{i}=rtrue+v;  %actual measurement (based on state)

         H=[];
         for el=k:-1:1,
            if range(el)==0,
               Ha=[zeros(3,6)];
            elseif pos{el}(1)^2==range(el)^2*cos(elev(el))^2,
               Ha=[transpose(pos{el})/range(el) 0 0 0;...
                     pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
                     pos{el}(2)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
                     -(1/range(el)-pos{el}(3)^2/range(el)^3)/cos(elev(el)) 0 0 0;...
                     0 0 0 0 0 0];
            elseif cos(elev(el))==0,
               Ha=[transpose(pos{el})/range(el) 0 0 0;zeros(2,6)];
            else
               Ha=[transpose(pos{el})/range(el) 0 0 0;...
                     pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
                     pos{el}(2)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
                     -(1/range(el)-pos{el}(3)^2/range(el)^3)/cos(elev(el)) 0 0 0;...
                     (1/range(el)/cos(elev(el))-pos{el}(1)^2/range(el)^3/cos(elev(el))-
pos{el}(1)^2*pos{el}(3)^2/range(el)^5/cos(elev(el))^3)/...
                     (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 ...
                     (-pos{el}(1)*pos{el}(2)/range(el)^3/cos(elev(el))-
pos{el}(1)*pos{el}(2)*pos{el}(3)^2/range(el)^5/cos(elev(el))^3)/...
                     (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 ...
                     (-pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el))-.5*pos{el}(1)*(-
2*pos{el}(3)/range(el)^2+2*pos{el}(3)^3/range(el)^4)/range(el)/cos(elev(el))^3)/...
                     (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 0 0 0];
            end
            H(3*el-2:3*el,6*el-5:6*el)=Ha;
         end

         Pneg=Ad*Ppos*transpose(Ad)+Qd;
         L=Pneg*transpose(H)*inv(R+H*Pneg*transpose(H));
```

```
      stateest1{i}=stateest1{i}+L*(Y{i}-rbar);
      Ppos=Pneg-L*H*Pneg;

      for j=k:-1:1,
         sigmax{i}(j)=sqrt(Ppos(6*j-5,6*j-5));
         sigmay{i}(j)=sqrt(Ppos(6*j-4,6*j-4));
         sigmaz{i}(j)=sqrt(Ppos(6*j-3,6*j-3));
         sigmaxdot{i}(j)=sqrt(Ppos(6*j-2,6*j-2));
         sigmaydot{i}(j)=sqrt(Ppos(6*j-1,6*j-1));
         sigmazdot{i}(j)=sqrt(Ppos(6*j,6*j));
      end
      i
   end;
end;

u1{i}=-Kd*[zeros(1,6*k)]';

t=1:dT:60*hours;

%Convert u, state, and stater to matrices for plotting
for i=1:dT:60*hours,
   for j=1:3*k,
      um(i,j)=u1{i}(j);
   end
   for j=1:6*k,
      statem(i,j)=state1{i}(j);
      staterm(i,j)=stater1{i}(j);
      statedifm(i,j)=statedif1{i}(j);
      stateestm(i,j)=stateest1{i}(j);
   end
for j=1:k,
   sigmaxm(i,j)=sigmax{i}(j);
   sigmaym(i,j)=sigmay{i}(j);
   sigmazm(i,j)=sigmaz{i}(j);
   sigmaxdotm(i,j)=sigmaxdot{i}(j);
   sigmaydotm(i,j)=sigmaydot{i}(j);
   sigmazdotm(i,j)=sigmazdot{i}(j);
end
```

```
end
%Need to make vectors same length for plotting when dT /= 1
for i=1:dT:60*hours,
   for el=1:dT-1,
      statem(i+el,:)=statem(i,:);
      staterm(i+el,:)=staterm(i,:);
      statedifm(i+el,:)=statedifm(i,:);
      stateestm(i+el,:)=stateestm(i,:);
      um(i+el,:)=um(i,:)*0;%No control when a maneuver not performed
   end
end

%Total Delta V
for i=1:k,
   dvx(i)=sum(abs(um(:,3*i-2)))*dT/1440;
   dvy(i)=sum(abs(um(:,3*i-1)))*dT/1440;
   dvz(i)=sum(abs(um(:,3*i)))*dT/1440;
   totdv(i)=dvx(i)+dvy(i)+dvz(i);     %km/day
end
totdv2=totdv*1000/86400;      %m/sec

%Position Error at end
for i=1:k,
   xerror(i)=statedifm(end,6*i-5);
   yerror(i)=statedifm(end,6*i-4);
   zerror(i)=statedifm(end,6*i-3);
   error(i)=sqrt(xerror(i)^2+yerror(i)^2+zerror(i)^2);
end

mi1=550;
mi2=100;
Isp1=10000;
Isp2=10000;

totdv2
disp('in m/s');
massprop2(1)=mi1*(1-exp(-totdv2(1)/Isp1/9.81));
for i=2:k,
```

```
   massprop2(i)=mi2*(1-exp(-totdv2(i)/Isp2/9.81));
end
massprop2*1000  %g
disp('in g');

error*1000
disp('in m');


%PLOTS
%true positions
color=['b','r','g','m','c','y'];
color=[color,color,color,color,color,color];
figure;
subplot(2,2,4),
for i=k:-1:2,
   plot3(statem(end,6*i-5),statem(end,6*i-4),statem(end,6*i-3),'or');
   hold on;
   plot3(statem(1,6*i-5),statem(1,6*i-4),statem(1,6*i-3),'om');
end
plot3(0,0,0,'*k');
grid on;hold off;xlabel('x (km)');ylabel('y (km)');zlabel('z (km)');

subplot(2,2,1),
for i=k:-1:2,
   plot(statem(end,6*i-5),statem(end,6*i-4),'or');
   hold on;
   plot(statem(1,6*i-5),statem(1,6*i-4),'om');
end
plot(0,0,'*k');title('Formation');
xlabel('x (km)');ylabel('y (km)');grid on;hold off;

subplot(2,2,2),
for i=k:-1:2,
   plot(statem(end,6*i-5),statem(end,6*i-3),'or');
   hold on;
   plot(statem(1,6*i-5),statem(1,6*i-3),'om');
end
```

```
plot(0,0,'*k');
xlabel('x (km)');ylabel('z (km)');grid on;hold off;

subplot(2,2,3),
for i=k:-1:2,
   plot(statem(end,6*i-4),statem(end,6*i-3),'or');
   hold on;
   plot(statem(1,6*i-4),statem(1,6*i-3),'om');
end
plot(0,0,'*k');
xlabel('y (km)');ylabel('z (km)');grid on;hold off;

%hub tracking errors
figure;
subplot(3,2,1),plot(t,statedifm(:,1));ylabel('x error (km)');grid on;
title('hub tracking error');
subplot(3,2,3),plot(t,statedifm(:,2));ylabel('y error (km)');grid on;
subplot(3,2,5),plot(t,statedifm(:,3));ylabel('z error (km)');grid on;xlabel('time');
subplot(3,2,2),plot(t,statedifm(:,4)*1000/86400);ylabel('xdot error (m/s)');grid on;
subplot(3,2,4),plot(t,statedifm(:,5)*1000/86400);ylabel('ydot error (m/s)');grid on;
subplot(3,2,6),plot(t,statedifm(:,6)*1000/86400);ylabel('zdot error (m/s)');grid on;xlabel('time');

%hub estimation errors
figure;
subplot(3,2,1),plot(t,statedifm(:,1)-stateestm(:,1),t,3*sigmaxm(:,1),'r',t,-3*sigmaxm(:,1),'r');
ylabel('x estimation error (km)');grid on;title('hub estimation error');
subplot(3,2,3),plot(t,statedifm(:,2)-stateestm(:,2),t,3*sigmaym(:,1),'r',t,-3*sigmaym(:,1),'r');
ylabel('y estimation error (km)');grid on;
subplot(3,2,5),plot(t,statedifm(:,3)-stateestm(:,3),t,3*sigmazm(:,1),'r',t,-3*sigmazm(:,1),'r');
ylabel('z estimation error (km)');xlabel('time');grid on;
subplot(3,2,2),plot(t,(statedifm(:,4)-stateestm(:,4))*1000/86400,t,3*sigmaxdotm(:,1)*1000/86400,'r',t,-
3*sigmaxdotm(:,1)*1000/86400,'r');
ylabel('xdot estimation error (m/s)');grid on;
subplot(3,2,4),plot(t,(statedifm(:,5)-stateestm(:,5))*1000/86400,t,3*sigmaydotm(:,1)*1000/86400,'r',t,-
3*sigmaydotm(:,1)*1000/86400,'r');
ylabel('ydot estimation error (m/s)');grid on;
subplot(3,2,6),plot(t,(statedifm(:,6)-stateestm(:,6))*1000/86400,t,3*sigmazdotm(:,1)*1000/86400,'r',t,-
3*sigmazdotm(:,1)*1000/86400,'r');
```

```
ylabel('zdot estimation error (m/s)');xlabel('time');grid on;

%hub satellite controls
figure;
subplot(3,1,1),plot(t,um(:,1));ylabel('control x');grid on;
title('hub control');
subplot(3,1,2),plot(t,um(:,2));ylabel('control y');grid on;
subplot(3,1,3),plot(t,um(:,3));ylabel('control z');xlabel('time');grid on;

if k > 1,
   for i=2:k,
      %drone tracking errors
      figure;
      subplot(3,2,1),plot(t,statedifm(:,6*i-5));ylabel('x error (km)');grid on;
      title(['drone ' num2str(i) ' tracking error']);
      subplot(3,2,3),plot(t,statedifm(:,6*i-4));ylabel('y error (km)');grid on;
      subplot(3,2,5),plot(t,statedifm(:,6*i-3));ylabel('z error (km)');grid on;xlabel('time');
      subplot(3,2,2),plot(t,statedifm(:,6*i-2)*1000/86400);ylabel('xdot error (m/s)');grid on;
      subplot(3,2,4),plot(t,statedifm(:,6*i-1)*1000/86400);ylabel('ydot error (m/s)');grid on;
      subplot(3,2,6),plot(t,statedifm(:,6*i)*1000/86400);ylabel('zdot error (m/s)');grid on;xlabel('time');

      %drone estimation errors
      figure;
      subplot(3,2,1),plot(t,statedifm(:,6*i-5)-stateestm(:,6*i-5),t,3*sigmaxm(:,i),'r',t,-
3*sigmaxm(:,i),'r');
      ylabel('x estimation error (km)');grid on;title(['drone ' num2str(i) ' estimation error']);
      subplot(3,2,3),plot(t,statedifm(:,6*i-4)-stateestm(:,6*i-4),t,3*sigmaym(:,i),'r',t,-
3*sigmaym(:,i),'r');
      ylabel('y estimation error (km)');grid on;
      subplot(3,2,5),plot(t,statedifm(:,6*i-3)-stateestm(:,6*i-3),t,3*sigmazm(:,i),'r',t,-
3*sigmazm(:,i),'r');
      ylabel('z estimation error (km)');xlabel('time');grid on;
      subplot(3,2,2),plot(t,(statedifm(:,6*i-2)-stateestm(:,6*i-
2))*1000/86400,t,3*sigmaxdotm(:,i)*1000/86400,'r',t,-3*sigmaxdotm(:,i)*1000/86400,'r');
      ylabel('xdot estimation error (m/s)');grid on;
      subplot(3,2,4),plot(t,(statedifm(:,6*i-1)-stateestm(:,6*i-
1))*1000/86400,t,3*sigmaydotm(:,i)*1000/86400,'r',t,-3*sigmaydotm(:,i)*1000/86400,'r');
      ylabel('ydot estimation error (m/s)');grid on;
```

```
      subplot(3,2,6),plot(t,(statedifm(:,6*i)-
stateestm(:,6*i))*1000/86400,t,3*sigmazdotm(:,i)*1000/86400,'r',t,-3*sigmazdotm(:,i)*1000/86400,'r');
      ylabel('zdot estimation error (m/s)');xlabel('time');grid on;

      %drone controls
      figure;
      subplot(3,1,1),plot(t,um(:,3*i-2));ylabel('control x');grid on;
      title(['drone ' num2str(i) ' control']);
      subplot(3,1,2),plot(t,um(:,3*i-1));ylabel('control y');grid on;
      subplot(3,1,3),plot(t,um(:,3*i));ylabel('control z');xlabel('time');grid on;
   end
end
```

Appendix E: Formation Reorientation Matlab Simulation

Formchange.m

```
%FORMATION REORIENTATION SIMULATION
clear all
close all

%NUMBER OF SATELLITES
k=31;
%PLACE IN LISSAJOUS ORBIT (1-358)
place=2;
%MANEUVER INTERVAL
dT=1; %(minutes)
%NUMBER OF HOURS TO RUN
hours=24;
%FOCAL LENGTH
fl=0.5; %km  (.5 or 4 for SI)

rad=pi/180;

getA;
getref;

Aj(:,:)=Agen{place}(:,:);%time in seconds
Aj(1:3,1:3)=Aj(1:3,1:3)*86400;%convert to days
Aj(4:6,1:3)=Aj(4:6,1:3)*86400^2;
Aj(4:6,4:6)=Aj(4:6,4:6)*86400;

statera=stater{place};
staterb=stater{place+1};

syms tt ti
T=1;
%SYSTEM DYNAMICS AND COST WEIGHTING
disp('integrating');
```

```
Bj=[zeros(3);eye(3)];

Phij1=eye(6)+Aj*ti+Aj^2*ti^2/2;
Bdj=Phij1*Bj;
Bdj=int(Bdj,ti,0,tt);

Phij=eye(6)+Aj*dT/1440+Aj^2*(dT/1440)^2/2;
%WEIGHTING MATRICES
Wj=[eye(3) zeros(3);zeros(3) eye(3)*.001]*1e6;
Vj=eye(3);

Wdj=transpose(Phij)*Wj*(Phij);
Wdj=int(Wdj,tt,0,dT/1440);

Vdj=(transpose(Bdj)*Wj*Bdj+Vj);
Vdj=int(Vdj,tt,0,dT/1440);

Mdj=transpose(Phij)*Wj*Bdj;
Mdj=int(Mdj,tt,0,dT/1440);

%PROCESS NOISE
qnode=1e-24;
Qj=[zeros(3) zeros(3);zeros(3) eye(3)*qnode];
Qdj=int(Phij*Qj*transpose(Phij),tt,0,dT/1440);

tt=dT/1440;
%DISCRETE PARTS
Adj=Phij;%state transition matrix
Bdj=eval(Bdj); %discrete control mapping matrix
Wdj=eval(Wdj); %discrete state weighting matrix
Vdj=eval(Vdj); %discrete control weighting matrix
Mdj=eval(Mdj); %disrete state-control couple weight
Qdj=eval(Qdj); %discrete process noise matrix
for j=k:-1:1,
   A(6*j-5:6*j,6*j-5:6*j)=Aj;
   Ad(6*j-5:6*j,6*j-5:6*j)=Adj;
   Wd(6*j-5:6*j,6*j-5:6*j)=Wdj;
   Vd(3*j-2:3*j,3*j-2:3*j)=Vdj;
```

```
   Md(6*j-5:6*j,3*j-2:3*j)=Mdj;
   Qd(6*j-5:6*j,6*j-5:6*j)=qnode*Qdj;
   Q(6*j-5:6*j,6*j-5:6*j)=Qj;
   if j==1,
      Bd(6*j-5:6*j,3*j-2:3*j)=Bdj;
   else,
      Bd(6*j-5:6*j,3*j-2:3*j)=Bdj;
      Bd(6*j-5:6*j,1:3)=-1*Bdj;
   end
end

disp('calculating gain');
Kd=dlqr(Ad,Bd,Wd,Vd,Md); %discrete control gain

%MEASUREMENTS
R1=diag([.1 .1*3/1500000 .1*3/1500000].^2);
Rj=diag([.0001 .0001*3/f1 .0001*3/f1].^2);
R(1:3,1:3)=R1;
for j=k:-1:2,
   R(3*j-2:3*j,3*j-2:3*j)=Rj;
end

Pnot1=diag([1 1 1 86.4 86.4 86.4].^2);
Pnotj=diag([.001 .001 .001 .0864 .0864 .0864].^2);Pnot(1:6,1:6)=Pnot1;
for j=k:-1:2,
   Pnot(6*j-5:6*j,6*j-5:6*j)=Pnotj;
end

%DEFINE initial GEOMETRY (spherical coordinates)
for i=2:k,
   dr(i)=2*f1;%all radial components
end
for i=2:16,
   dphi(i)=(3*pi/2+asin(.25/dr(2))/15*(17-i));
end
for i=17:k,
   dphi(i)=(3*pi/2-asin(.25/dr(2))/15*(i-16));
end
```

```
dtheta(2)=-asin(.25/dr(2))/15*15;
dtheta(3)=asin(.25/dr(2))/15*10;
dtheta(4)=-asin(.25/dr(2))/15*14;
dtheta(5)=asin(.25/dr(2))/15*4;
dtheta(6)=asin(.25/dr(2))/15*6;
dtheta(7)=asin(.25/dr(2))/15*11;
dtheta(8)=asin(.25/dr(2))/15*14;
dtheta(9)=-asin(.25/dr(2))/15*3;
dtheta(10)=-asin(.25/dr(2))/15*13;
dtheta(11)=-asin(.25/dr(2))/15*1;
dtheta(12)=asin(.25/dr(2))/15*9;
dtheta(13)=asin(.25/dr(2))/15*5;
dtheta(14)=-asin(.25/dr(2))/15*4;
dtheta(15)=asin(.25/dr(2))/15*8;
dtheta(16)=asin(.25/dr(2))/15*7;
dtheta(17)=-asin(.25/dr(2))/15*9;
dtheta(18)=-asin(.25/dr(2))/15*8;
dtheta(19)=asin(.25/dr(2))/15*12;
dtheta(20)=-asin(.25/dr(2))/15*11;
dtheta(21)=-asin(.25/dr(2))/15*7;
dtheta(22)=asin(.25/dr(2))/15*15;
dtheta(23)=asin(.25/dr(2))/15*3;
dtheta(24)=asin(.25/dr(2))/15*13;
dtheta(25)=-asin(.25/dr(2))/15*2;
dtheta(26)=-asin(.25/dr(2))/15*5;
dtheta(27)=-asin(.25/dr(2))/15*10;
dtheta(28)=-asin(.25/dr(2))/15*12;
dtheta(29)=asin(.25/dr(2))/15*2;
dtheta(30)=-asin(.25/dr(2))/15*6;
dtheta(31)=asin(.25/dr(2))/15*1;

dphicent=3*pi/2;
dthetacent=0;

%convert to cartesian
dycent=dr(2)*sin(dphicent)*cos(dthetacent);
dxcent=dr(2)*sin(dthetacent);
dzcent=dr(2)*cos(dphicent)*cos(dthetacent);
```

```
for i=2:k,
   dyj(i)=dr(i)*sin(dphi(i))*cos(dtheta(i));
   dxj(i)=dr(i)*sin(dtheta(i));
   dzj(i)=dr(i)*cos(dphi(i))*cos(dtheta(i));

   dy(i)=dyj(i)-dycent/2;
   dx(i)=dxj(i)-dxcent/2;
   dz(i)=dzj(i)-dzcent/2;
end

IC=[];
for i=1:k,
   if i==1,
      dx(i)=statera(1);
      dy(i)=statera(2);
      dz(i)=statera(3);
      dxdot(i)=statera(4);
      dydot(i)=statera(5);
      dzdot(i)=statera(6);
      IC=[IC;dx(i);dy(i);dz(i);dxdot(i);dydot(i);dzdot(i)];
   else,
      IC=[IC;dx(i);dy(i);dz(i);0;0;0];
   end
end

%Define reference geometry
dx1(1)=statera(1);%hub
dy1(1)=statera(2);
dz1(1)=statera(3);

for i = 2:k,
   dr(i)=2*fl;
   dphi1(i)=-dtheta(i)+3*pi/2;
   dtheta1(i)=dphi(i)-3*pi/2;
end

dthetacent=0;
```

```
dphicent=3*pi/2;

%convert to cartesian
for i=2:k,
   dyj1(i)=dr(i)*sin(dphi1(i))*cos(dtheta1(i));
   dxj1(i)=dr(i)*sin(dtheta1(i));
   dzj1(i)=dr(i)*cos(dphi1(i))*cos(dtheta1(i));

   dycent=dr(i)*sin(dphicent)*cos(dthetacent);
   dxcent=dr(i)*sin(dthetacent);
   dzcent=dr(i)*cos(dphicent)*cos(dthetacent);

   dy1(i)=dyj1(i)-dycent/2;
   dx1(i)=dxj1(i)-dxcent/2;
   dz1(i)=dzj1(i)-dzcent/2;
end


ref1=[];
for i=1:k,
   if i==1,
      dxdot1(i)=statera(4);
      dydot1(i)=statera(5);
      dzdot1(i)=statera(6);
      ref1=[ref1;dx1(i);dy1(i);dz1(i);dxdot1(i);dydot1(i);dzdot1(i)];
   else,
      ref1=[ref1;dx1(i);dy1(i);dz1(i);0;0;0];
   end
end

%SIMULATE

for i=1:dT:60*hours,
   if i == 1,
      state1{i}=IC;
      stater1{i}=ref1;
      statedif1{i}=state1{i}-stater1{i};
      stateest1{i}=statedif1{i};
```

```
   Ppos=Pnot;
   for j=k:-1:1,
      sigmax{i}(j)=sqrt(Ppos(6*j-5,6*j-5));
      sigmay{i}(j)=sqrt(Ppos(6*j-4,6*j-4));
      sigmaz{i}(j)=sqrt(Ppos(6*j-3,6*j-3));
      sigmaxdot{i}(j)=sqrt(Ppos(6*j-2,6*j-2));
      sigmaydot{i}(j)=sqrt(Ppos(6*j-1,6*j-1));
      sigmazdot{i}(j)=sqrt(Ppos(6*j,6*j));
   end
   i
else
   w=sqrt(Q/T)*randn(length(Q),1);
   u1{i-dT}=-Kd*(stateest1{i-dT});

   stater1{i}=stater1{i-dT}; %just for sizing
   stater1{i}(1:6)=(staterb(1:6)-stater1{1}(1:6))/(60*hours/dT)*i+stater1{1}(1:6);

   statedif1{i}=Ad*statedif1{i-dT}+Bd*u1{i-dT}+w;
   stateest1{i}=Ad*stateest1{i-dT}+Bd*u1{i-dT};

   statebar1{i}=stateest1{i}+stater1{i};
   state1{i}=statedif1{i}+stater1{i};

   v=chol(R)'*randn(length(R),1);

   rtrue=[];
   rbar=[];
   for el=1:k,
      postrue{el}=state1{i}(6*el-5:6*el-3);
      rangetrue(el)=(transpose(postrue{el})*postrue{el})^.5;
      elevtrue(el)=asin(-postrue{el}(3)/rangetrue(el));
      aztrue(el)=asin(postrue{el}(1)/rangetrue(el)/cos(elevtrue(el)));

      pos{el}=statebar1{i}(6*el-5:6*el-3);
      range(el)=(transpose(pos{el})*pos{el})^.5;
      elev(el)=asin(-pos{el}(3)/range(el));
      az(el)=asin(pos{el}(1)/range(el)/cos(elev(el)));
```

```
      rtrue=[rtrue;rangetrue(el);elevtrue(el);aztrue(el)];
      rbar=[rbar;range(el);elev(el);az(el)];
   end

   Y{i}=rtrue+v;  %actual measurement (based on state)

   H=[];
   for el=k:-1:1,
      if range(el)==0,
         Ha=[zeros(3,6)];
      elseif pos{el}(1)^2==range(el)^2*cos(elev(el))^2,
         Ha=[transpose(pos{el})/range(el) 0 0 0;...
               pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               pos{el}(2)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               -(1/range(el)-pos{el}(3)^2/range(el)^3)/cos(elev(el)) 0 0 0;...
               0 0 0 0 0 0];
      elseif cos(elev(el))==0,
         Ha=[transpose(pos{el})/range(el) 0 0 0;zeros(2,6)];
      else
         Ha=[transpose(pos{el})/range(el) 0 0 0;...
               pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               pos{el}(2)*pos{el}(3)/range(el)^3/cos(elev(el)) ...
               -(1/range(el)-pos{el}(3)^2/range(el)^3)/cos(elev(el)) 0 0 0;...
               (1/range(el)/cos(elev(el))-pos{el}(1)^2/range(el)^3/cos(elev(el))-
pos{el}(1)^2*pos{el}(3)^2/range(el)^5/cos(elev(el))^3)/...
               (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 ...
               (-pos{el}(1)*pos{el}(2)/range(el)^3/cos(elev(el))-
pos{el}(1)*pos{el}(2)*pos{el}(3)^2/range(el)^5/cos(elev(el))^3)/...
               (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 ...
               (-pos{el}(1)*pos{el}(3)/range(el)^3/cos(elev(el))-.5*pos{el}(1)*(-
2*pos{el}(3)/range(el)^2+2*pos{el}(3)^3/range(el)^4)/range(el)/cos(elev(el))^3)/...
               (1-pos{el}(1)^2/range(el)^2/cos(elev(el))^2)^.5 0 0 0];
      end
      H(3*el-2:3*el,6*el-5:6*el)=Ha;
   end

   Pneg=Ad*Ppos*transpose(Ad)+Qd;
   L=Pneg*transpose(H)*inv(R+H*Pneg*transpose(H));
```

```
      stateest1{i}=stateest1{i}+L*(Y{i}-rbar);
      Ppos=Pneg-L*H*Pneg;

      for j=k:-1:1,
         sigmax{i}(j)=sqrt(Ppos(6*j-5,6*j-5));
         sigmay{i}(j)=sqrt(Ppos(6*j-4,6*j-4));
         sigmaz{i}(j)=sqrt(Ppos(6*j-3,6*j-3));
         sigmaxdot{i}(j)=sqrt(Ppos(6*j-2,6*j-2));
         sigmaydot{i}(j)=sqrt(Ppos(6*j-1,6*j-1));
         sigmazdot{i}(j)=sqrt(Ppos(6*j,6*j));
      end

      i
   end;
end;

u1{i}=-Kd*[zeros(1,6*k)]';

t=1:dT:60*hours;

%Convert u, state, and stater to matrices for plotting
for i=1:dT:60*hours,
   for j=1:3*k,
      um(i,j)=u1{i}(j);
   end
   for j=1:6*k,
      statem(i,j)=state1{i}(j);
      staterm(i,j)=stater1{i}(j);
      statedifm(i,j)=statedif1{i}(j);
      stateestm(i,j)=stateest1{i}(j);
   end
   for j=1:k,
      sigmaxm(i,j)=sigmax{i}(j);
      sigmaym(i,j)=sigmay{i}(j);
      sigmazm(i,j)=sigmaz{i}(j);
      sigmaxdotm(i,j)=sigmaxdot{i}(j);
      sigmaydotm(i,j)=sigmaydot{i}(j);
      sigmazdotm(i,j)=sigmazdot{i}(j);
```

```
   end
end
%Need to make vectors same length for plotting when dT /= 1
for i=1:dT:60*hours,
   for el=1:dT-1,
      statem(i+el,:)=statem(i,:);
      staterm(i+el,:)=staterm(i,:);
      statedifm(i+el,:)=statedifm(i,:);
      stateestm(i+el,:)=stateestm(i,:);
      um(i+el,:)=um(i,:)*0;%No control when a maneuver not performed
   end
end

%Total Delta V
for i=1:k,
   dvx(i)=sum(abs(um(:,3*i-2)));
   dvy(i)=sum(abs(um(:,3*i-1)));
   dvz(i)=sum(abs(um(:,3*i)));
   totdv(i)=dvx(i)+dvy(i)+dvz(i);     %km/day^2
end
totdv1=totdv*1000/86400*dT/1440;      %m/sec

%Position Error at end
for i=1:k,
   xerror(i)=statedifm(end,6*i-5);
   yerror(i)=statedifm(end,6*i-4);
   zerror(i)=statedifm(end,6*i-3);
   error(i)=sqrt(xerror(i)^2+yerror(i)^2+zerror(i)^2);
end

mi1=550;
mi2=100;
Isp1=10000;
Isp2=10000;

totdv1
disp('in m/s');
massprop1(1)=mi1*(1-exp(-totdv1(1)/Isp1/9.81));
```

```
for i=2:k,
   massprop1(i)=mi2*(1-exp(-totdv1(i)/Isp2/9.81));
end
massprop1*1000  %g
disp('in g');

error*1000
disp('in m');

%PLOTS
%true positions
color=['b','r','g','m','c','y'];
color=[color,color,color,color,color,color];
figure;
subplot(2,2,4),
for i=k:-1:2,
   plot3(statem(end,6*i-5),statem(end,6*i-4),statem(end,6*i-3),'or');
   hold on;
   plot3(statem(1,6*i-5),statem(1,6*i-4),statem(1,6*i-3),'om');
end
plot3(0,0,0,'*k');
grid on;hold off;xlabel('x');ylabel('y');zlabel('z');title('true positions (close)');

subplot(2,2,1),
for i=k:-1:2,
   plot(statem(end,6*i-5),statem(end,6*i-4),'or');
   hold on;
   plot(statem(1,6*i-5),statem(1,6*i-4),'om');
end
plot(0,0,'*k');
xlabel('x');ylabel('y');grid on;hold off;

subplot(2,2,2),
for i=k:-1:2,
   plot(statem(end,6*i-5),statem(end,6*i-3),'or');
   hold on;
   plot(statem(1,6*i-5),statem(1,6*i-3),'om');
end
```

```
plot(0,0,'*k');
xlabel('x');ylabel('z');grid on;hold off;

subplot(2,2,3),
for i=k:-1:2,
   plot(statem(end,6*i-4),statem(end,6*i-3),'or');
   hold on;
   plot(statem(1,6*i-4),statem(1,6*i-3),'om');
end
plot(0,0,'*k');
xlabel('y');ylabel('z');grid on;hold off;

%hub tracking errors
figure;
subplot(3,2,1),plot(t,statedifm(:,1));ylabel('x error (km)');grid on;
title('hub tracking error');
subplot(3,2,3),plot(t,statedifm(:,2));ylabel('y error (km)');grid on;
subplot(3,2,5),plot(t,statedifm(:,3));ylabel('z error (km)');grid on;xlabel('time');
subplot(3,2,2),plot(t,statedifm(:,4)*1000/86400);ylabel('xdot error (m/s)');grid on;
subplot(3,2,4),plot(t,statedifm(:,5)*1000/86400);ylabel('ydot error (m/s)');grid on;
subplot(3,2,6),plot(t,statedifm(:,6)*1000/86400);ylabel('zdot error (m/s)');grid on;xlabel('time');

%hub estimation errors
figure;
subplot(3,2,1),plot(t,statedifm(:,1)-stateestm(:,1),t,3*sigmaxm(:,1),'r',t,-3*sigmaxm(:,1),'r');
ylabel('x estimation error (km)');grid on;title('hub estimation error');
subplot(3,2,3),plot(t,statedifm(:,2)-stateestm(:,2),t,3*sigmaym(:,1),'r',t,-3*sigmaym(:,1),'r');
ylabel('y estimation error (km)');grid on;
subplot(3,2,5),plot(t,statedifm(:,3)-stateestm(:,3),t,3*sigmazm(:,1),'r',t,-3*sigmazm(:,1),'r');
ylabel('z estimation error (km)');xlabel('time');grid on;
subplot(3,2,2),plot(t,(statedifm(:,4)-stateestm(:,4))*1000/86400,t,3*sigmaxdotm(:,1)*1000/86400,'r',t,-
3*sigmaxdotm(:,1)*1000/86400,'r');
ylabel('xdot estimation error (m/s)');grid on;
subplot(3,2,4),plot(t,(statedifm(:,5)-stateestm(:,5))*1000/86400,t,3*sigmaydotm(:,1)*1000/86400,'r',t,-
3*sigmaydotm(:,1)*1000/86400,'r');
ylabel('ydot estimation error (m/s)');grid on;
subplot(3,2,6),plot(t,(statedifm(:,6)-stateestm(:,6))*1000/86400,t,3*sigmazdotm(:,1)*1000/86400,'r',t,-
3*sigmazdotm(:,1)*1000/86400,'r');
```

```
ylabel('zdot estimation error (m/s)');xlabel('time');grid on;

%hub satellite controls
figure;
subplot(3,1,1),plot(t,um(:,1));ylabel('control x');grid on;
title('hub control');
subplot(3,1,2),plot(t,um(:,2));ylabel('control y');grid on;
subplot(3,1,3),plot(t,um(:,3));ylabel('control z');xlabel('time');grid on;

if k > 1,
   for i=2:k,
      %drone tracking errors
      figure;
      subplot(3,2,1),plot(t,statedifm(:,6*i-5));ylabel('x error (km)');grid on;
      title(['drone ' num2str(i) ' tracking error']);
      subplot(3,2,3),plot(t,statedifm(:,6*i-4));ylabel('y error (km)');grid on;
      subplot(3,2,5),plot(t,statedifm(:,6*i-3));ylabel('z error (km)');grid on;xlabel('time');
      subplot(3,2,2),plot(t,statedifm(:,6*i-2)*1000/86400);ylabel('xdot error (m/s)');grid on;
      subplot(3,2,4),plot(t,statedifm(:,6*i-1)*1000/86400);ylabel('ydot error (m/s)');grid on;
      subplot(3,2,6),plot(t,statedifm(:,6*i)*1000/86400);ylabel('zdot error (m/s)');grid on;xlabel('time');

      %drone estimation errors
      figure;
      subplot(3,2,1),plot(t,statedifm(:,6*i-5)-stateestm(:,6*i-5),t,3*sigmaxm(:,i),'r',t,-
3*sigmaxm(:,i),'r');
      ylabel('x estimation error (km)');grid on;title(['drone ' num2str(i) ' estimation error']);
      subplot(3,2,3),plot(t,statedifm(:,6*i-4)-stateestm(:,6*i-4),t,3*sigmaym(:,i),'r',t,-
3*sigmaym(:,i),'r');
      ylabel('y estimation error (km)');grid on;
      subplot(3,2,5),plot(t,statedifm(:,6*i-3)-stateestm(:,6*i-3),t,3*sigmazm(:,i),'r',t,-
3*sigmazm(:,i),'r');
      ylabel('z estimation error (km)');xlabel('time');grid on;
      subplot(3,2,2),plot(t,(statedifm(:,6*i-2)-stateestm(:,6*i-
2))*1000/86400,t,3*sigmaxdotm(:,i)*1000/86400,'r',t,-3*sigmaxdotm(:,i)*1000/86400,'r');
      ylabel('xdot estimation error (m/s)');grid on;
      subplot(3,2,4),plot(t,(statedifm(:,6*i-1)-stateestm(:,6*i-
1))*1000/86400,t,3*sigmaydotm(:,i)*1000/86400,'r',t,-3*sigmaydotm(:,i)*1000/86400,'r');
      ylabel('ydot estimation error (m/s)');grid on;
```

```
      subplot(3,2,6),plot(t,(statedifm(:,6*i)-
stateestm(:,6*i))*1000/86400,t,3*sigmazdotm(:,i)*1000/86400,'r',t,-3*sigmazdotm(:,i)*1000/86400,'r');
      ylabel('zdot estimation error (m/s)');xlabel('time');grid on;

      %drone controls
      figure;
      subplot(3,1,1),plot(t,um(:,3*i-2));ylabel('control x');grid on;
      title(['drone ' num2str(i) ' control']);
      subplot(3,1,2),plot(t,um(:,3*i-1));ylabel('control y');grid on;
      subplot(3,1,3),plot(t,um(:,3*i));ylabel('control z');xlabel('time');grid on;
   end
end
```

Title: Formation Flying Satellite Control Around the L2 Sun-Earth Libration Point

Author: Nicholas H. Hamilton

Rank/Service: Second Lieutenant, United States Air Force

Year: 2001

Number of pages: 165

Degree: Master of Science

Institution: The School of Engineering and Applied Science of the George Washington University

Abstract:

A growing interest in formation flying satellites demands development and analysis of control and estimation algorithms for station-keeping and formation maneuvering. This thesis discusses the development of a discrete linear-quadratic-regulator control algorithm for formations in the vicinity of the L2 sun-earth libration point. The development of an appropriate Kalman filter is included as well. Simulations are created for the analysis of the station-keeping and various formation maneuvers of the Stellar Imager mission. The simulations provide tracking error, estimation error, and control effort results. From the control effort, useful design parameters such as ΔV and propellant mass are determined. For formation maneuvering, the drone spacecraft track to within 4 meters of their desired position and within 1.5 millimeters per second of their desired zero velocity. The filter, with few exceptions, keeps the estimation errors within their three-sigma values. Without noise, the controller performs extremely well, with the drones tracking to within several micrometers. Each drone uses around 1 to 2 grams of propellant per maneuver, depending on the circumstances.

Bibliography:

1. Szebehely, V., *Theory of Orbits: The Restricted Problem of Three Bodies.* Academic Press, Inc. 1967.
2. Barden, B.T., and Howell, K.C., "Fundamental Motions Near Collinear Libration Points and Their Transitions," *Journal of the Astronautical Sciences*, Vol. 46. 1998.
3. Howell, K.C., Barden, B.T., and Lo, M.W., "Application of Dynamical Systems Theory to Trajectory Design for a Libration Point Mission," *Journal of the Astronautical Sciences*, Vol. 45. 1997.
4. Farquhar, R., *The Control and Use of Libration-Point Satellites.* NASA Technical Report R-346. 1970.
5. Gomez, G., Masdemont, J., and Simo, C., "Quasihalo Orbits Associated with Libration Points," *Journal of the Astronautical Sciences*, Vol. 46. 1998.
6. Scheeres, D.J., and Vinh, N.X., "Dynamics and Control of Relative Motion in an Unstable Orbit," AIAA Paper 2000-4135. 2000.

7. Hoffman, D.A., *Station-keeping at the Collinear Equilibrium Points of the Earth-Moon System*, JSC-26189. NASA Johnson Space Center. 1993.
8. "Microwave Anisotropy Probe," http://map.gsfc.nasa.gov. October, 2001.
9. "The GSFC Stellar Imager Homepage," http://hires.gsfc.nasa.gov/~si/. October, 2001.
10. "Micro Arcsecond X-Ray Imaging Mission," http://maxim.gsfc.nasa.gov. October, 2001.
11. "MAXIM Pathfinder," http://maxim.gsfc.nasa.gov/pathfinder.html. October, 2001.
12. "Terrestrial Planet Finder," http://tpf.jpl.nasa.gov. October, 2001.
13. Speyer, J.L., "Computation and Transmission Requirements for a Decentralized Linear-Quadratic-Gaussian Control Problem." *IEEE Transactions on Automatic Control*; AC-24(2). 1979.
14. Carpenter, J.R., "A Preliminary Investigation of Decentralized Control for Satellite Formations." Proceedings of the 2000 IEEE Aerospace Conference, March 18-25, 2000.
15. Carpenter, J.R., "Decentralized Control of Satellite Formations." To appear in *International Journal of Robust and Nonlinear Control.*
16. Carpenter, J.R., Folta, D.C., and Quinn, D.A., "Integration of Decentralized Linear-Quadratic-Gaussian Control into GSFC's Universal 3-D Autonomous Formation Flying Algorithm." AIAA Paper 99-4269, AIAA Guidance Navigation &Control, Modeling & Simulation Technologies and Atmospheric Flight Mechanics Conference and Exhibit, August 9-11, 1999.
17. "Magnetospheric Constellation," http://stp.gsfc.nasa.gov/missions/mc/mc.htm. October 2001.
18. "Laser Interferometer Space Antenna," http://lisa.jpl.nasa.gov. October 2001.
19. Wie, B., *Space Vehicle Dynamics and Control.* AIAA Education Series. 1998.
20. Nise, N., *Control Systems Engineering.* 2nd Edition. Addison-Wesley Publishing Company. 1995.
21. Friedland, B., *Control System Design.* McGraw-Hill. 1986.
22. Stengel, R.F., *Optimal Control and Estimation.* Dover Publications, Inc. 1994.
23. Phillips, C.L., and Nagle, H.T., *Digital Control System Analysis and Design.* 3rd Edition. Prentice Hall, Inc. 1995.
24. Bryson, A.E, Jr., and Ho, Y.C., *Applied Optimal Control: Optimization, Estimation, and Control.* Hemisphere Publishing Corporation. 1975.
25. Wagner, C., "Formation Flying Around Sun-Earth Libration Point L1." International Space University. 2000.
26. Brown, R.C., and Hwang, P.Y.C., *Introduction to Random Signals and Applied Kalman Filtering.* 2nd Edition. John Wiley & Sons, Inc. 1992.
27. Maybeck, P.S., *Stochastic Models, Estimation, and Control.* Volume 1. Academic Press, Inc. 1979.
28. Carpenter, K.G., Neff, S.G., Schrijver, C.J., Allen, R.J., and Rajagopal, J., "The Stellar Imager (SI) Mission Concept." In proceedings of the 36th Liege Astrophysical Colloquium: From Optical to Millimetric Intefereometry: Scientific and Technical Challenges. 2001.
29. Carpenter, K.G., Lyon, R.G., Schrijver, C.J., Mundy, L.J., Allen, R.J., and Rajagopal, J., "Imaging the Surfaces and Interiors of Other Stars: The Stellar Imager (SI) Mission Concept." 2001.